University of Paderborn

Self-Organizing Ad-hoc Mobile Robotic Networks

Emi Mathews

Dissertation

in Computer Science

A thesis submitted to the

Faculty of Electrical Engineering, Computer Science, and Mathematics

of the

University of Paderborn

in partial fulfillment of the requirements for the degree of

doctor rerum naturalium (Dr. rer. nat.)

Paderborn, Germany

July 12, 2012

Supervisors:

Prof. Dr. rer. nat. Franz Josef Rammig, University of Paderborn

Prof. Dr. math. Friedhelm Meyer auf der Heide, University of Paderborn

Date of public examination: 24. August 2012

Acknowledgments

I can no other answer make, but, thanks, and thanks.

William Shakespeare

I would like to sincerely thank Prof. Dr. Franz Rammig for his constant guidance and support during my research. He always encouraged my ideas and provided great freedom to carry out research in my area of interest. The scientific opportunities I got through his encouragement to publish papers and attend international conferences, helped me to grow professionally. Despite his busy schedules, he was always available for discussions and set apart time to review my work. I have learned a lot from him especially on how to think positively and optimistically. Words cannot express my gratitude to him.

I am deeply indebted to Jun. Prof. Hannes Frey for the countless discussions I had with him. It helped me to sharpen my ideas. His reviews helped me to publish my works in very reputed conferences.

I would like to extend special thanks to Prof. Dr. Friedhelm Meyer auf der Heide, the member of my committee and my second reviewer. I would also like to specially thank other members of my committee, Prof. Dr. Ulrich Rückert and Dr. Michael Thies.

I also wish to thank whole heartily all the members of the working group and the International Graduate School for supporting my work. I am grateful for the discussions and joint work with my colleagues especially Yara Khaluf. I am also grateful to Peter Janacik, Fahad Bin Tariq, Marcio F. S. Oliveira, Jakob family and all other friends for making the life enjoyable in Paderborn. I enjoyed playing soccer with C-Lab robots and as well as C-Lab colleagues.

I wish to thank my students, especially Ciby Mathew, Tobias Graf and Suranga Kulathunga who helped in formalizing some of the algorithms as well as their implementations. The painstaking setup procedures for practical multiple robots experiments were made easy with the support of Rakesh Garimala and Andry Tanoto.

I specially thank my family members especially my parents, in-laws and sisters for their unconditional love and encouragement. My parents dedicated their entire life for my education and deserve more credit than I could ever give them. I heartily thank my wife Mabel for the enormous support during my work and the careful often late-night reviews of my papers and this dissertation. I give my deepest gratitude and love for her support and encouragement that made this

dissertation possible. I thank our baby daughter Joanna for jumping on lap to play the nursery rhymes on my computer while I write my dissertation and remind me of the important things in life.

Finally I am grateful to God for showering His love and affection and for giving me the wonderful opportunity to come to Germany and pursue my research.

Abstract

Ad-hoc Mobile Robotic Network (AMRoNet) consists of a large collection of mobile robots which creates a wireless network for collaborative and co-operative task accomplishment. Technical advancement and miniaturization of computing, communicating, sensing and actuating devices enabled building small and low cost robots such as swarm robots and Micro Air Vehicles (MAV) for creating such networks. Self-organization is a key feature that makes AMRoNets adaptive and useful in many application scenarios such as sensing and monitoring and urban search and rescue. Self-organizing systems interact locally according to simple rules and the global behavior of the system emerges from these local interactions. Self-configuration, self-healing and self-optimization are three key features of self-organizing networks.

In AMRoNets, self-configuration is achieved through self-deployment which enables the nodes to start from a relatively compact initial configuration and spread out in an area, maximizing the coverage and keeping the network connected, using simple local rules. Self-healing mechanisms allow automatic reconfiguration of the network in case of failures. Coverage requirements vary with applications. We consider two types of coverage maximization problems: sensing range-based and communication range-based area coverage. For sensing range-based area coverage problems, we introduce a new swarm-based algorithm based on the local rules, namely separation, cohesion and alignment, used in modeling of fish schooling. Empirical analysis shows that it outperforms most prominent state-of-the-art algorithms by achieving better and faster coverage. The algorithm could also be used for communication range-based area coverage problems, but it is not optimal in such cases. Hence a greedy deployment algorithm which could achieve a near optimal performance in obtaining communication range-based area coverage and useful in certain application scenarios has also been proposed.

Self-optimization capability enables the network to adjust regularly and route efficiently in large-scale networks, especially when topology changes are frequent. We concentrate on the routing aspect of the self-optimizing networks. Routing algorithm which uses only local rules or local information is the primary choice for self-organizing AMRoNets. Geographic routing is one such algorithm which is very useful in AMRoNets because of its simplicity, scalability and low routing overhead. The basic geographic routing uses a greedy forwarding step and a planar graph based FACE routing step, whenever packets cannot be forwarded according

to the greedy step. FACE routing guarantees message delivery if it is applied on a planar embedding of the communication network. Challenges in obtaining a planar graph in real wireless networks with irregular radio range and imprecise node location information make this algorithm inapplicable in practical scenarios. To solve this problem, we propose a new localized planarization algorithm based on a topological cluster-based overlay graph construction. To make the overlay graphs planar, a new explicit planarization technique has been applied. Empirical analysis shows that this overlay graph-based macroscopic planarization approach is location fault tolerant and produces planar graphs in most of the realistic wireless networks. Thus the highly efficient geographic routing algorithm is now applicable in practical scenarios and self- optimizing communication is feasible in AMRoNets.

Contents

List of Figures

List of Algorithms

CHAPTER 1

Introduction

The real voyage of discovery lies not in seeking new landscapes, but in having new eyes.

M. Proust (1871-1922)

Ad-hoc Mobile Robotic Network (AMRoNet) consists of a large collection of mobile robots which create a wireless network for collaborative and co-operative task accomplishment. Technical advancement and miniaturization of computing, communicating, sensing and actuating devices enabled building small and low cost robots such as swarm robots [47] and Micro Air Vehicles (MAV) [77] for creating such networks. A good example for an AMRoNet is the Mobile Wireless Sensor Network (MWSN) where mobility of the nodes is used to improve coverage, connectivity, and lifetime of the network. The key difference between AMRoNet and a standard Mobile Ad-hoc Network (MANET) is that, the position and motion of a node in MANETs are determined by its owner and cannot be controlled by other nodes; whereas in AMRoNets, these properties are controllable from other nodes in the network.

AMRoNets will facilitate many existing and new application areas. We classify them into two based on the type of service they provide:

1. Sensing and monitoring: In sensing and monitoring applications, they require special sensors to monitor ambient conditions such as temperature, pressure, humidity, stress and movement. Many potential applications include hazard detection, disaster relief, environmental control, biodiversity mapping, surveillance, reconnaissance, space exploration etc. Typically Wireless Sensor Networks (WSN) are employed for such applications [14]. Replac-

ing WSN nodes with AMRoNet nodes eases the deployment phase. Nodes being mobile are capable of deploying themselves and also compensate the shortcomings in the deployment process by moving to better positions in post deployment phases. In sparse networks, when nodes die, mobile nodes can relocate to connect the lost or weak communication pathways. Moreover, in some applications which require precise deployment for proper functioning, but too dangerous or inaccessible for external agents to perform the deployment, nodes deploying themselves are the best solution. An example is using mobile nodes to sense and measure poisonous gas or nuclear leakage.

2. Communication infrastructure: In scenarios such as urban search and rescue, due to the aftermath of natural or man-made disasters such as earthquakes, tsunamis, hurricanes, wars or explosions, the fixed communication infrastructure that could support communication between rescue agents are often destroyed. In other scenarios such as battlefields or exploration of unknown terrains, e.g. subterranea or remote planets, no such infrastructure to support communication exist. There are also scenarios where there is a sudden increase in network usage, e.g. a large demonstration or large crowd assembling in a disaster area, such that the existing infrastructure cannot provide the expected quality of service. In these scenarios, AMRoNets could be deployed to act as a temporary infrastructure to facilitate communication, due to its flexibility in deployment, faster setup time and in many cases cost effectiveness.

Although AMRoNet can be used in many other applications, we focus on the above two classes of applications which bear the following key properties:

- Uses the mobility of the nodes for self-deployment or for network optimization purposes

- Network is fairly static during the operational mode

Traditional multi-robot systems are centralized. Though they can process greater amounts of data and achieve more accuracy or even better performances in certain cases, they often suffer from the common pitfalls, such as poor scalability, single point of failure and high complexity. AMRoNets are envisioned to work in environments where there is no infrastructure for centralized administration or support. Hence purely distributed strategies are more suitable for AMRoNets.

Self-organization is a key feature seen in biological world of cells, organisms, and groups. Self-organizing systems interact locally according to simple rules and the global behavior of the system emerges from these local interactions. The best examples are the social insects such as ants, though not very intelligent on an individual level, perform complex tasks such as nest building, brood care, food foraging, etc. through simple local interactions. Inspired from nature, we adopt these principles of self-organization in the design of AMRoNets. It makes the system adaptive and flexible to the application requirements. Key features of such systems are best described by the term Self-X: Self-configuration, Self-healing and Self-optimization.

Self-configuration is the capability of the nodes to deploy themselves and configure networks automatically without any centralized administration or support, fulfilling application requirements. The networks are expected to work in the absence of any fixed infrastructure. In AMRoNets, self-configuration is achieved through self-deployment which enables the nodes to start from a relatively compact initial configuration and spread out in an area maximizing *coverage* using simple local rules. Coverage requirements vary with applications. An important aspect to consider while maximizing the coverage is to keep the network of robots *connected*. Thus the problem of *coverage* maximization maintaining the *connectivity*, denoted as $C-C$, arises when the robots spread out. In this dissertation, we study this problem from various applications' point of view and propose algorithms for solving it.

Self-healing mechanisms aim at reducing the impact of failure in network functionalities due to inoperative nodes or environmental changes by allowing automatic reconfiguration of the network. In AMRoNets, whenever the coverage or connectivity is lost, the self-healing actions are performed by the nodes automatically which then relocate the nodes to quickly regain coverage and connectivity. We focus on designing algorithms that are capable of recovering from failures easily.

In a data communication network, if two nodes are not connected directly by a communication link, messages between them need to be forwarded by intermediate nodes. Finding a path between two nodes through which to send messages is a fundamental problem called *routing* [167]. In traditional networks, there are dedicated routers or access points to relay the messages. However in ad-hoc networks the nodes themselves act as routers. This means that whenever nodes which are not in the direct wireless transmission range want to communicate, they sent

messages to the intermediate nodes in their range, which then forward the message further until the message reaches the destination, if there is a path between the sender and receiver nodes.

The decentralized nature of AMRoNet routing makes them suitable for a variety of applications, but it introduces many new challenges compared to the traditional wired or managed (infrastructure) wireless networks. One important challenge is the unavailability of a persistent routing table in ad-hoc networks especially when topology changes frequently. Efficient routing in such scenarios is non-trivial.

We envision our AMRoNet as a self-optimizing network which is capable of routing efficiently in large-scale networks with frequent topology changes. We concentrate only on this important aspect of the self-optimizing networks in our work. A large number of efficient routing protocols are available that could support self-optimized routing feature, but none of them are directly applicable for AMRoNets because of the following two challenges involved in the AMRoNet routing scenarios:

- Irregular wireless range
- Imprecise location information of the nodes

In this dissertation we propose new techniques and algorithms which address these two challenges and make self-optimized routing feasible in AMRoNets.

1.1 Coverage and Connectivity

We have seen that self-configuration in AMRoNet is achieved through self-deployment. Coverage and connectivity are the two fundamental requirements in AMRoNet self-deployment process. Finding an optimal self-deployment strategy that would minimize cost, reduce computation and communication overhead, be resilient to node failures, and provide a high degree of coverage with network connectivity is extremely challenging [74].

Coverage requirements vary with applications. In some cases a regular deployment is needed whereas in some other cases a random deployment is acceptable. In some cases a high degree of coverage with many nodes covering a particular point is important whereas in some other cases a low degree of coverage is sufficient. E.g. a military surveillance application require a high degree of coverage where as a low degree of coverage is sufficient for animal habitat monitoring.

Historically, three types of coverages have been defined by Gage [67] for problems such as detection of targets in a surveillance area:

- Blanket coverage - to achieve a static arrangement of elements that maximizes the detection rate of targets appearing within the coverage area.
- Barrier coverage - to achieve a static arrangement of elements that minimizes the probability of undetected penetration through the barrier.
- Sweep coverage - to move a group of elements across a coverage area in a manner which addresses a specified balance between maximizing the number of detections per time and minimizing the number of missed detections per area. (A sweep is roughly equivalent to a moving barrier.)

Our applications focus mainly on blanket coverage according to the above taxonomy, where the main objective is to maximize the total covered area. We look at two different area coverage problems in our work: sensing range-based area coverage and communication range-based area coverage.

1.1.1 Sensing range-based coverage

Each sensor has a physical sensing range within which it is able to detect or measure a physical property. Sensing range-based coverage tells us how well each point in the sensing area is covered by the sensors. It can be considered as a measure of quality of service.

In many AMRoNet applications, especially sensing and monitoring, sensing range-based coverage is important. Nodes use their locomotive ability to self-deploy in such areas to maximize the total covered sensing area. Various factors such as environmental geography, presence of obstacles and resource constraints affect the quality of coverage in such self-deployment applications. An additional constraint in our applications is the network connectivity maintenance during the self-deployment process.

In this dissertation, we focus on designing a bio-inspired algorithm for achieving optimal sensing range based coverage with connectivity. Bio-inspired algorithms work well in environments where prior-knowledge about the environment is minimal. They adapt to unforeseen changes in the task environment quickly. Hence, they are the most suitable choice for our applications. We have designed a new bio-inspired algorithm referred to as *swarm algorithm* for achieving coverage and connectivity. The main idea used in our swarm algorithm is born from

the schooling behavior of fish which "optimizes" the sensing range-based coverage with connectivity constraint naturally. We look at this natural model to design our algorithm.

1.1.2 Communication range-based coverage

The second type of area coverage problem is *communication range-based coverage*, where the objective is to maximize the total communication area covered. In many AMRonet applications, especially in those where they act as a temporary infrastructure to facilitate communication, it is no longer the *sensing-range based coverage* which is relevant; rather the focus is on communication-range based coverage.

The additional constraint in these applications, i.e. keeping the network connected, makes communication range-based coverage maximization problem different from the sensing range-based problem, as the communication range is much larger compared to the physical sensing range. In such problems, the swarm-based algorithm is a good solution, but not optimal if the objective is to minimize the total number of nodes used to cover the given area. Hence we have introduced a localized greedy deployment approach for such scenarios.

1.2 Efficient Routing in AMRoNets

We have seen that self-optimization capability enables the network to adjust regularly and route efficiently in large-scale networks, especially when topology changes frequently. Routing algorithm which uses only local rules or local information is the primary choice for self-organizing AMRoNets. Geographic routing is one such algorithm which is very useful in AMRoNets because of its simplicity, scalability and low routing overhead.

The basic geographic routing approach works by greedily forwarding the messages to nodes which minimize a local forwarding metric, such as *distance to the destination*. For all greedy routing variants, forwarding might end up at a node which is the best compared to its neighbor nodes [34]. However, from these nodes the messages cannot be forwarded further in greedy mode according to the forwarding metric, though a path from source to destination may exist.

So far, FACE routing, originally described in [26], is the only known localized single path approach to recover from such greedy forwarding failures. FACE

routing uses localized right/left hand graph traversal, to find a path around the boundaries of the void regions. FACE routing guarantees message delivery if it is applied on a planar embedding of the communication network. A planar embedding is a graph representation in a plane where edges intersect only at their end points [34]. In general, a wireless network topology is not a planar embedding. Hence, a planarization step prior to routing is required.

Planarization methods typically construct planar subgraphs by removing links from the original network topology. Distributed and localized planarization is difficult for real wireless networks [99]. Existing localized planarization techniques [26,69,100], i.e., methods where each node is required to know only its one (or two) hop neighbors, work well for Unit Disk Graphs (UDG) [40]. They do not work correctly in realistic wireless networks, which in most cases do not obey the UDG property. In such network graphs, the planarization causes network partitions, unidirectional links and intersecting links.

In addition, localized planarization methods assume that the exact node position information is at hand. However in practice, geographic location information is not exact. Location inaccuracies degrade the performance of localized planarization algorithms. They cause incorrect edge removal during planarization and may disconnect the planar subgraph. Even small errors can lead to incorrect routing decisions with noticeable performance degradation [102, 155].

Thus, we have real wireless networks with two problems: irregular radio range and node location inaccuracy, which affect localized planarization algorithm badly and limits the use of geographic routing in AMRoNets. In our work, we solve this problem by proposing a new localized planarization algorithm which is location fault tolerant and produces planar graphs in most realistic wireless networks.

1.3 Contributions

Key contributions of this dissertation are the algorithms listed below:

a) Sensing range-based coverage: For AMRonet applications that require sensing range-based coverage, we designed a swarm-based coverage and connectivity maintenance algorithm. The algorithm is based on the local rules used by fish while schooling. Each robot is subject to three forces: i) A *separation force* that pushes it away from its neighbours and increases the size of the swarm. ii) A *cohesion force* that maintains the connectivity of the swarm. iii) An *alignment force* that keeps it aligned to its neighbours and makes relocation faster. Empiri-

cal analysis shows that this swarm-based algorithm outperforms most prominent state-of-the-art algorithms by achieving better and faster coverage.

b) Communication range-based coverage: For AMRonet applications that require communication range-based coverage, we introduced a localized agent-assisted router deployment approach. The algorithm uses a greedy deployment strategy for deploying routers effectively into the area maximizing coverage and a triangular deployment strategy to connect different connected components of routers from different base stations. Empirical analysis shows that the proposed algorithm is a very efficient localized approach to create AMRoNets.

c) Localized Link Removal and Addition based Planarization algorithm: All existing localized planarization techniques are implicit planarization, which means that the links are removed irrespective of the fact whether an intersection exists or not. They provably produce planar graphs in restricted wireless models such as UDG. However in graphs that are not UDG, these approaches do not work, as they cause disconnections or fail to create planar graphs. We propose an explicit planarization algorithm called Localized Link Removal and Addition based Planarization (LLRAP) algorithm which solves these problems in graphs which are more generic than UDG. It removes links only when it detects an intersection in its local neighborhood.

d) Localized planarization algorithm for realistic wireless networks: None of the localized planarization algorithms including LLRAP are capable of planarizing arbitrary realistic wireless graphs. Their performance becomes worse when there are location errors. We proposed a localized planarization algorithm based on overlay graph creation that creates planar graphs in realistic wireless models and which is location fault tolerant. The planarization algorithm creates an overlay graph by topology-based clustering. Using a cross link detection and repair algorithm, the intersections in the overlay graph are removed locally. This localized planarization algorithm planarizes all networks used in the simulation study.

In addition to the algorithms, our contributions also include new tools as well as extensions to the existing tools to make multi-robot simulations and real robot experiments easier and empirical analysis feasible. This includes a new framework for simplifying multi-robot simulations and their evaluations as well as extensions to our robot development environment and network simulator.

1.4 Organization of the Dissertation

The rest of the dissertation is organized as follows:

In Chapter 2, we focus on the resources and platforms used for creating AMRoNets and the new tools and extensions to conduct multi-robot simulations and real robot experiments.

In Chapter 3, we focus on the self-deployment aspect of nodes with an objective of maximizing area coverage with network connectivity and the self-relocation aspect which maintains network connectivity in case of failures. We look at two different area coverage problems: sensing range-based and communication range-based and provide the theoretical background on the optimal coverage in both cases. We also provide a list of most prominent algorithms related to the self-deployment and self-relocation problems in this chapter.

In Chapter 4, we focus on the sensing range-based coverage problem and design a new bio-inspired algorithm to solve the problem efficiently. In Chapter 5, our focus is on the communication range-based coverage problem and we propose an agent-assisted router deployment to effectively deploy router nodes. We adopted an empirical analysis based evaluation of these algorithms in various scenarios to show its effectiveness. Simulations as well as real robot experiments are conducted for the empirical analysis.

We concentrate on the routing aspect of the self-optimizing networks in Chapter 6 and look at various routing protocols which are useful for AMRoNet routing. Considering the distributed, low overhead and self-configuring routing aspects, geographic routing is found to be an interesting solution. We describe about the basic components of geographic routing and the most relevant geographic routing protocols in this chapter.

FACE routing is a geographic routing protocol which is especially interesting due to its guaranteed delivery in plane graphs. We look at the various planarization algorithms to create plane graphs and their limitations in Chapter 7 and also propose a new planarization algorithms that solves some of these problems.

We look at the practical aspects of the geographic routing in real AMRoNet scenarios, identifies the main challenges and the impact of these challenges on the performance of geographic routing and various solutions proposed to overcome them in Chapter 8. One of the key challenges identified is localized planarization in real wireless networks. A new algorithm addressing this challenge is presented in Chapter 9, which is found to be effective in most test cases.

Chapter 10 summarizes our work and describes open questions and possible future directions.

CHAPTER 2

Resources and Platforms

In this chapter we focus on the resources and platforms used for creating AMRoNets. Small and low costs robots such as Swarm-bots, MAVs, MSNs or other mini-robots are suitable candidates for AMRoNets. In our work, we choose the mini-robot Bebot [83] developed at our institute (Heinz Nixdorf Institute, University of Paderborn) as the candidate for AMRoNet. Programming the robots, to instruct them about their tasks and to control their actions, is a non-trivial problem. Many software platforms and frameworks have been proposed to make programming robots easier [23]. In robotics, simulators are highly recommended regardless of whether an actual robot is available or not. We see the details of the robotic platform and the simulator used for programming Bebot and our own contributions such as tool chains, framework and various extensions we added, explained in this chapter.

To predict the behavior of an ad hoc network and evaluate the performance of communication protocols, typically network simulators are used. We also use a network simulator called Shox [118] developed in our working group for simulating AMRoNets. We see the basic features of the simulator and the packages we added to improve the simulation experience in this chapter.

2.1 Bebot robot

Bebot robot depicted in Figure 2.1 has a size of $9 \times 9 \times 5\ cm^3$ with 12 infrared sensors (sender-receiver pairs) and their controllers mounted directly on the chassis using Molded Interconnect Device (MID) technology [83]. The actuation consists of a chain drive. The robot has two slots for extension board. The lower extension

board with an ARM 7 micro-controller is responsible for motor control and power supply. It also contains a three axis acceleration sensor, a yaw rate gyroscope and a sensor for battery monitoring. The upper board is equipped with a low

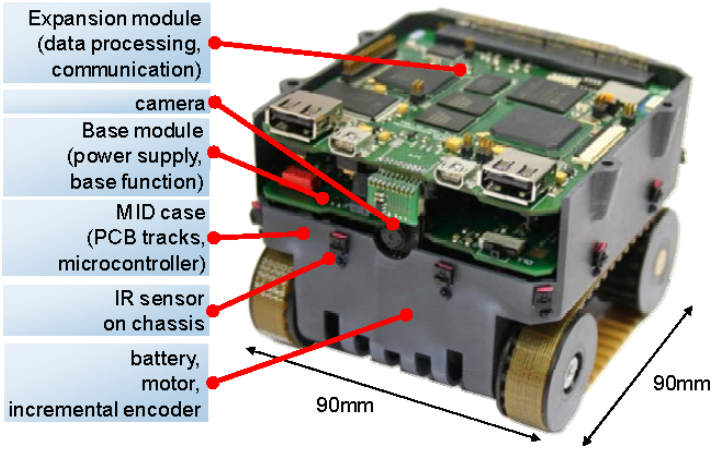


Figure 2.1: Bebot mini-robot

power 520 MHz Marvel PXA 270 processor, 64 MB main memory and 64 MB flash memory. It also has a Field Programmable Gate Array (FPGA) that allows complex algorithm to be efficiently executed in the hardware. It supports ZigBee, Bluetooth and WLAN standards for communication and provides additional interfaces, such as USB, MMC/SD card, audio, LCD and camera modules.

2.2 Player/Stage robotic framework

Player/Stage robotic framework [73] is one of the most popular robotic platforms used in research. We use this platform to develop control programs for the Bebot robots. The basic parts of the framework are 1) the network server called *Player*, which could be considered as a robotic hardware abstraction layer for robotic devices and 2) the *Stage*, which is a two-dimensional simulator for mobile robots. The robot control program acts as a client and talks to the Player server over a TCP socket.

2.2.1 Player

Three fundamental parts of the Player server are interfaces, drivers and devices as shown Figure 2.2.

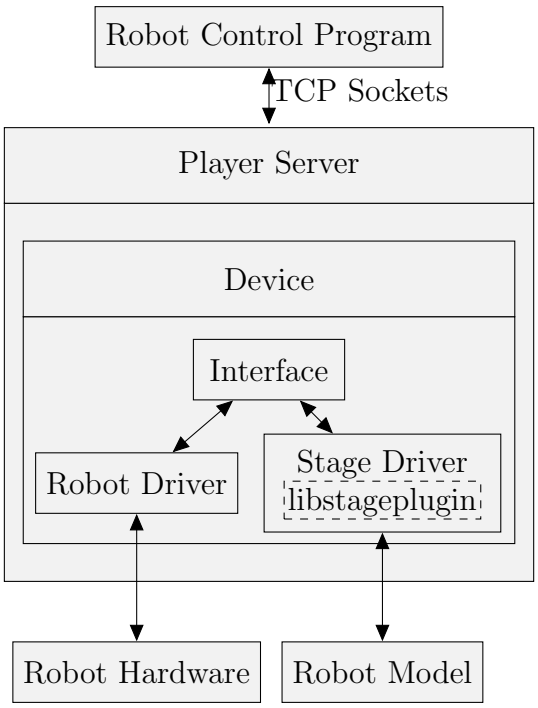


Figure 2.2: Player Stage

Interfaces define the syntax of how to interact with a certain class of hardware such as sensors or actuators. It provides a format in which data or meta-data can be transmitted from a sensor or to an actuator. An example is the *position2d* interface [73] that handles robots' positions in 2D, which basically interacts with the odometry sensors and motors of the robot.

The driver is a software that communicates with the actual hardware and it is written specifically for each hardware type and model. It translates the inputs and outputs to conform to one or more relevant interfaces. An example is the *sicklms200* driver [73] which communicates with the SICK LMS200, retrieve range data and convert the SICK-specific format to the format defined by the ranger interface. Player also allows virtual drivers which encapsulate useful algorithms in a way that they can be easily reused. A virtual driver usually gets data from other drivers using their interfaces (instead of getting them from the real hardware), process the data using advanced algorithms, and provide the results using their interfaces. An example of a virtual driver is the *acml* driver based on the Adaptive Monte-Carlo Localization algorithm [55] which supports the *position2d* interface and a more sophisticated *localize* interface [72].

A device is the topmost abstraction in Player which is basically a driver that is bound to an interface so that Player can communicate to it directly. The fully-

qualified device address consists of 5 parts [145]: *key:host:robot:interface:index*, where *key* is a unique name, *host* and *robot* are usually the host name and TCP port on which the Player server listens, *interface* is the interface name of the device, and *index* is an interface specific sequence number as more than once device can support the same interface.

Player has a rich set of drivers for commonly used robots and robotic related hardware. However a driver for Bebot was not available "out of the box". Player has a framework that supports writing drivers for new hardware. A new driver may either be built-in at compile time or plugged-in at runtime, depending on whether it is included in the next version of Player or maintained as a separate package. New plugin drivers, such as *libbebotmotor* providing *position2d* interface, *libbebotir* providing *ir* interface and *libcamerav4l2* providing camera interface has been developed for the Bebot robots [83].

2.2.2 Configuration file

To provide access for client control programs, the Player server has to be started with a configuration file initially. Player server needs this configuration file which informs it about the drivers to initialize along with definitions and settings for each driver. Driver settings are specific to each driver and must contain the *name* of the driver and a list of the devices the driver *provides*, and for virtual drivers, a list of devices they *require* at startup [145].

The configuration file is composed of one or more driver sections, each of which instantiates and configures a single driver. An example of a configuration file is given below:

```
driver
(
 name "bebotmotor"
 provides ["position2d:0"]
 plugin "Player/libbebotmotor"
 device "/sys/bus/i2c/devices/0-0058/speed"
)
```

The *name* and *provides* parameters are mandatory information and they indicate which driver to use and what kind of information is received from the driver. Player instantiate the *bebotmotor* driver physically connected to the hardware device over the I2C port. It provides a *position2d* device under the address

position2d:0, where no key is declared, host and robot are implicitly defined as *localhost* and *6665*. Hence they are not explicitly mentioned in the configuration file.

2.2.3 Client program

Player allows user control programs to be written in high level languages such as C, C++, Java or Python that supports TCP sockets by providing client-side utilities and libraries. Client libraries are built on a *service proxy* model in which clients maintain local objects that are proxies for remote services [73]. There are two kinds of proxies: 1) a special server proxy *PlayerClient* to establish connection with the Player server and 2) the various device-specific proxies to subscribe to the respective devices. An example is given below:

```
PlayerClient client_name(hostname, port);
Position2dProxy positionProxy_name(&client_name,index);
RangerProxy rangerProxy_name(&client_name,index);
```

User programs first create a PlayerClient proxy object using a given host name and port number to connect to the server and use it to create the device specific proxies.

Now a read-think-act loop is implemented. It is the actual control program for the robot. In the loop, the client periodically processes incoming messages to update the proxy objects with information containing latest sensor data.

2.2.4 Stage

Stage is the two-dimensional multi-robot simulator aimed at providing fairly simple and computationally cheap models to control large population of robots without accessing real hardware and environments [73]. Thus it allows rapid development of controllers that will eventually drive real robots. Though it is usable as a stand-alone simulation library, together with Player it works as a driver which provides a simulator back-end. Figure 2.3 shows a simulated stage environment.

Stage has a configurable and composable device *models* which support various sensors and actuators. The basic model simulates an object with its body made up of a list of lines and with basic properties such as position, size, velocity, color and visibility to various sensors. These models are available through Player's standard interfaces. Hence from the client's point of view, there is no difference between

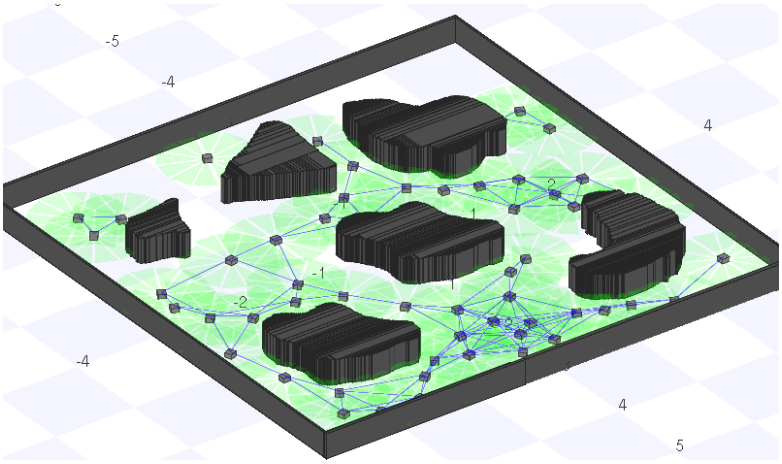


Figure 2.3: Stage simulation environment

the real robot devices and their simulated Stage equivalents. A sample Player configuration file is given below:

```
driver
(
  name "stage"
  provides [ "simulation:0" ]
  plugin "stageplugin"

  # load the named file into the simulator
  worldfile "bebot.world"
)
# Create a Stage driver and attach position2d, ranger and wifi
# interfaces to the model "bebot"
driver
(
  name "stage"
  provides [ "6665:position2d:0" "6665:ranger:0" "6665:wifi:0"]
  model "bebot"
)
```

which specifies that the *stage* driver is used by using the plugin *libstageplugin*. It also specifies interfaces the simulated robot model *bebot* provides, namely *position2d*, *ranger* and *wifi*. The world file describes the models and the environment which the Stage simulates. The length of the simulation, update cycle interval etc. can also be specified.

2.3 Framework for multi-robot simulations

Player client library allows multiple clients to communicate with player server. However there is no support for controlling multiple robots simultaneously in an efficient manner. We have created a framework that would simplify multiple robot control and simulation. It also helps to setup and run experiments quickly and very easily.

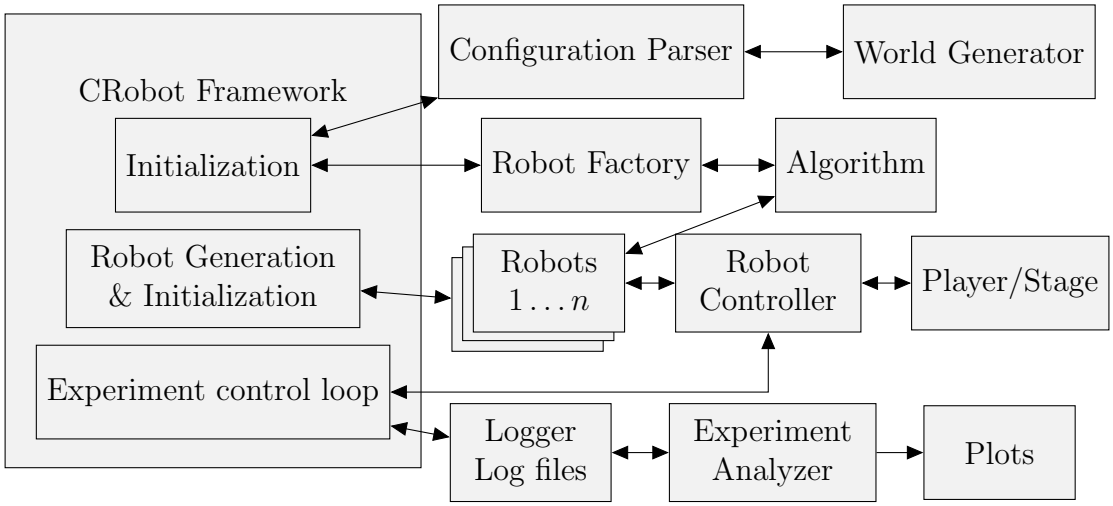


Figure 2.4: Framework for multi-robot simulations

The architecture of the framework is given in Figure 2.4. The basic components of the framework are:

2.3.1 World Generator

The World Generator creates the .world, .cfg and .inc files required by Player/Stage to run the robot simulator. These are not complex files but generating them automatically is helpful because Player/Stage does not natively support the dynamic creation of robots. On executing

```
./bebotgenerator [world] [config] [nrobots] [bitmap] [width]
 [height]  [comm_range] [seed]
```

specifying the world directory, configuration details for deployment such deployment mode (random or regular), region (inside base station or entire region), number of robots, bitmap of the environment (our institute's floor plan, cave or

empty region), width and height of the region, communication radius and seed for random generator, we get we get .world, .cfg and .inc files required by Player/Stage simulator. Examples files can be found in the Appendix. The `bebotgenerator` respects the terrain bitmaps and carefully places the robots in the free space. The generator could be easily extended for different experimental setups. One could also use existing configuration and world files available in the Player/Stage repository.

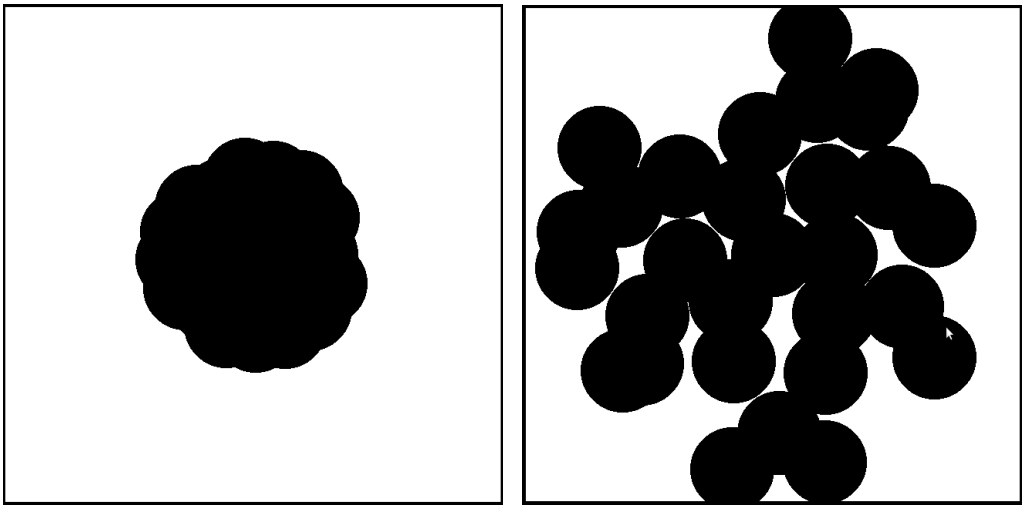
2.3.2 Robot control Generator

The *CRobot Framework* block has an initialization phase which uses a *Config. Parser* to parse the *world generator* files to identify the type and number (n) of robots. The *CRobot Framework* then creates n instances of $CRobots$ with the server proxy `PlayerClient` and device-specific proxies specifying `hostname` and `ports`. The *Robot Factory* block of the Framework setup the robot control algorithm such that during the robot initialization stage, the robot controller is dynamically linked to desired robot control logic.

The *CRobot Framework* then executes the main control loop which could be run in a threaded or sequential manner. Each robot controller is executed according to the control logic specified. The controller access the Player/Stage to get data from the drivers. It then update its position and neighbor list based on the data, create new goal point according to the control logic and write commands to the driver to navigate to the goal point avoiding obstacles. For behavior-based control, we have written a set of commonly used behaviors such as obstacle avoidance, wall following and goto goal. The Framework uses a logger to log the new position information and other details into a log file according to the sampling interval specified. When the maximum simulation time is reached the control loop is exited.

The user's control logic can be easily implemented by extending the CRobot class and implementing the onControl function.

```
class CRobotAlgorithm : public CRobot {
public:
CRobotAlgorithm();
virtual ~CRobotAlgorithm();
virtual void onControl();
};
```



(a) Initial Coverage **(b)** Final Coverage

Figure 2.5: Covered region of interest

```
void CRobotAlgorithm::onControl() {
.......
}
```

2.3.3 Experiment Analyzer

In our framework we have an experiment analyzer which contains a set of analysis and plotting tools. The *calmetric* tool takes the generated log file and find covered region of interest (ROI) at each point of time and generate an output file for the plotting tools. Figure 2.5 shows an example of the initial coverage and final coverage in a sample run. The plotting tool uses R language/environment [175] and gnuplot [179] for creating statistics and graphics from the log files created by the multiple independent runs.

2.4 Teleworkbench

Teleworkbench is a tele-operated platform or testbed for managing multi-robot experiments [168]. It allows remote users to set-up, execute, visualize and analyze their experiments over Internet. It is depicted in Figure 2.6. Key features of the Teleworkbench are:

- automatic environment building

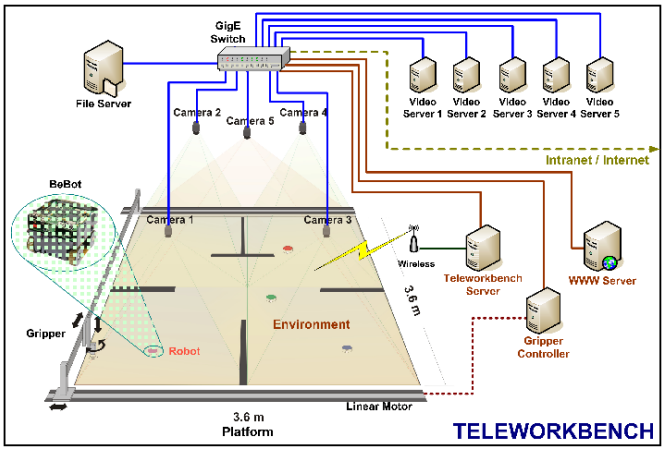


Figure 2.6: Teleworkbench environment

- remote execution of user programs
- integration with Player/Stage robotic framework
- data logging
- robot tracking (upto sixty-four robots)
- visualization and analysis tool

The Teleworkbench system has a field of size $3.6 \times 3.6\ m^2$ which could be divided into four sub-fields for parallel experiments. A 6-bit barcode-like marker placed on the top of robots allows the Video Server to identify the robots and helps in estimating their position and orientation in the field. The server also encodes the video and streams it to provide a live video of the experiment.

Robots can communicate with each other or with the Teleworkbench Server over Bluetooth. After the experiments, the server also generates a video with superimposed computer generated objects for better analysis of the experiments.

2.5 Bebot software development

2.5.1 Bebot development environment

The operating system which runs on Bebot robot is the Embedded Linux with a modified Linux kernel 2.6.24 [83].The software building is done via OpenEmbedded [113] development environment which allows the creation of a fully usable Linux operating system. OpenEmbedded is a collection of metadata used

to cross-compile, package and install software packages [113]. It handles cross-compilation and inter-package dependencies efficiently, creates images and feeds from packages of different formats such as tar, rpm, deb and ipk and supports many machines, distribution and architectures. It uses bitBake [61] tool to cross-compile and build packages. The OpenEmbedded software has been extended with an overlay called OpenRobotix to include robot specific information, patches and additional software like the Player network server and drivers for the robotic hardware. OpenRobotix generates operating system images, and the software development environment (SDE) for Bebot robot systems. The commands:

```
bitbake openrobotix-image
```

generates the root file system (rootfs) and the Linux image (uImage) and

```
bitbake meta-toolchain-openrobotix
```

generates the Cross Compiler Tool-chain of the software development environment.

User control programs use the Cross Compiler Tool-chain to generate executable code for ARM processors.

2.5.2 Framework

For making the evaluation of experiments easier, the algorithms tested on simulators need be run on the robots more or less without any change. A hardware framework is designed replacing the low level functionalities of the simulation framework with new features. One new feature is the use of an additional position proxy. As the position2d driver implemented on the lower board of the Bebot do not work properly, we had to rely on the Teleworkbench server to get an estimate of the current position of the robots. So we had two position2d proxies one for issuing motor commands for steering the robot and the other to communicate with the Teleworkbench to estimate the location information.

Another new feature is use of blackboard proxy for neighborhood estimation. In Stage simulations, Wi-Fi device is used to collect information about neighboring robots and the range of these devices can be easily adjusted in the configuration file. Using real Wi-Fi or Bluetooth devices in the Teleworkbench, all robots will be in the communication range of other robots and hence using Wi-Fi is not useful for multi-hop scenarios. To solve this problem, we use a filter at the server which filters the recent position information from the position log files in the server and write them on a `blackboard`. Bebot robots subscribe to blackboard and whenever

there is an update on the `blackboard`, they get the updated information. From this information, the framework then finds the neighbor information.

2.6 Network simulator

To evaluate the performance of communication protocols, typically network simulators are used. We use the Shox network simulator [118] developed in our working group for testing the protocols. Shox is a discrete-event simulator, where events such as node movements, packets, internal messages and timers are inserted into a global event queue (a priority queue) based on the delivery time. The delivery time specifies the time at which events have to be removed from the queue and delivered to the address where they have to be processed. An event has a unique ID and a timestamp which denotes the delivery time. The central manager called `SimulationManager` fetches events from the queue and delivers them to the appropriate target event handlers. The simulation time is not an absolute quantity but a virtual quantity which is updated to the value stored in the current event that is being processed. Thus, true parallel event handling is possible [118].

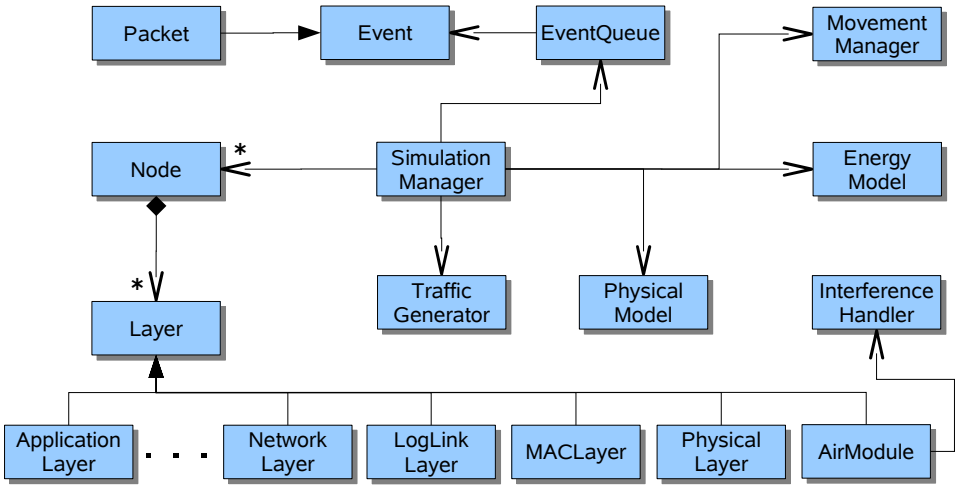


Figure 2.7: ShoX Architecture [118]

Architecture of ShoX is given in Figure 2.7. It follows the OSI layered architecture [167] with five among the seven OSI layers are present by default. Additional layers can be included easily at any point in the layer stack by deriving from the abstract super classes which specify the interfaces and functionalities. An example of simulation specific mac layer and network layer extension is given below:

```
public class MAC_IEEE802_11bg_DCF extends MACLayer{
...
}

public class AodvNetworkLayer extends NetworkLayer{
...
}
```

In addition to the basic OSI layers, an additional layer called `AirModule` which manages radio states of a node such as sending, receiving and idle listening as well as handles channel related issues such as signal interferences, is present. A packet arriving at a node is first processed by an interference handler which also gets notification from the `AirModule` regarding the interferences during the packet's transmission. Depending on the implementation logic of the handler and interference, the packet is discarded or forwarded to the physical layer of the receiver. The `PhysicalModel` compute whether the receiver is reachable; if not, whether it is in the interference range, based on the geographic positioning of the sending and receiving node and the estimated signal strength.

There is also a `TrafficGenerator` which support different traffic models and generates traces. Users can also use their own existing traffic trace files. For simulating node mobility, different mobility models are supported. `MovementManager` creates node movements according to the mobility model used. ShoX has an `EnergyManager` that uses the concept of devices which are basically special components of a node like the network interface card, the power manager, the CPU or attached sensors. Different devices can be registered as power suppliers or power consumers.

We have seen that `SimulationManager` fetches events from the `EventQueue` and processes them. Events can be of the following types: Simulation events, events for a node or events for a specific layer. Simulation events (e.g.movements) are used internally by the simulation and cannot be issued by the protocols being simulated. Events for a node (e.g. initialize) are directed to all layers of a node, whereas event for a layer (e.g. packets, `WakeupCall`) are dispatched to certain layers of a node. Packets are special events which are sent downwards in the layer stack until they reach the physical layer from which they are sent over the physical network to reach the target node and then go upward until they reach the destination layer. `WakeUpCall` is used to schedule future events in the system by setting a timeout and an event is issued when the timeout expires.

To run a simulation, the configuration details such as number of nodes, the size of the deployment area, the layer stack of the nodes, the signal propagation, mobility, and traffic models needs to be specified. Currently Shox uses a GUI to input the parameters either by reading from a configuration file or providing them manually. During the simulation, ShoX records all relevant events in a log file. Shox visualization unit plays back the recorded events. However, none of the controls provided such as pausing, step back or forward works correctly and visualization gets stuck. Parsing log files with large number of events need a long waiting time to begin the simulation.

We have designed a simple visualization tool for Shox which displays interesting events at run time. A snapshot of the visualization is shown in Figure 2.8, which

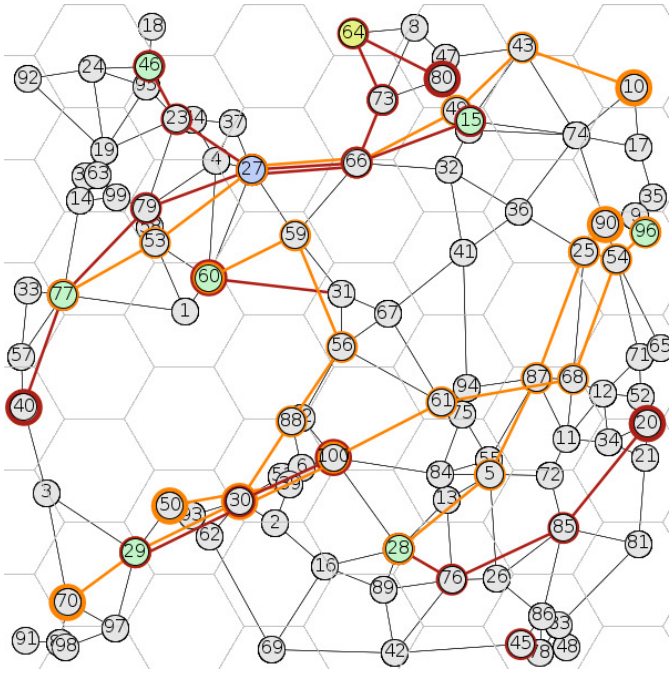


Figure 2.8: ShoX visualization

displays network topology and routing paths taken in specific routing protocol simulation. The visualization tool contains classes for drawing shapes such as points, lines, and curves, handling multiple drawing windows and saving drawings to a file in different formats. We extended the simulator by creating a set of utilities for handling geometric objects such as line, circle and hexagon and for serialization. A pool of algorithms, e.g Delaunay tessellation, BFS, Dijkstra, planarization and planarity testing algorithms and many state of the art network protocols; e.g. FACE routing and Hexagonal Cluster Routing were also added. New mobility

models; e.g. `TerrainRespectingRandomWaypoint` and tools for evaluation e.g. Plot were also added to the code base.

CHAPTER 3

Coverage and Connectivity in AMRoNets

Self-configuration enables the AMRoNet nodes to deploy themselves and configure networks automatically which is achieved mainly through self-deployment. It is especially useful in situations where a precise deployment is infeasible, but a desired amount of coverage is required for proper functioning. Self-deployment capability helps the nodes to achieve a configuration that maximizes area coverage. The coverage problem can be defined as placing minimum number of nodes in a given field such that every point is optimally covered. Achieving optimal coverage by self-deployment in presence of obstacles, noise and varying topography is extremely challenging. Keeping the network of robots connected is equally important while maximizing coverage, as a configuration that maximizes coverage with disconnected nodes is often not useful from the higher-level objective of the applications which deploy AMRoNets. Thus coverage and connectivity together can be considered as a quality of service in AMRoNet applications.

Self-healing mechanisms allow automatic reconfiguration of AMRoNets in case of failures. Whenever coverage or connectivity is lost, the self-reconfiguration process allows the nodes to relocate quickly to regain coverage and connectivity. This makes the system adaptive to hostile environmental conditions.

In this chapter we study the self-deployment and self-relocation problems.

3.1 Preliminaries

Let N denote the total number of robots and R denote an individual robot. The environment where the robots spread out is a 2-D area, denoted as A. Let $A_i = \pi r^2$ denote the coverage area of a robot R_i with r denoting its *coverage radius*.

The objective of our algorithm is to maximize the total area covered, where the *total coverage* is calculated as follows:

$$Coverage = \frac{\bigcup_{i=1}^{N} A_i}{A} \tag{3.1}$$

Degree of coverage, often denoted in literature as k-coverage, can be defined as the number of nodes that cover a particular point in the given area A [74]. We have seen in Chapter 1 that coverage requirements vary with applications and some applications require high degree of coverage. However, in this work we concentrate on those applications where 1-coverage is sufficient, i.e. at least one node covers any point in A.

We model the AMRoNet as a graph $G = (V, E)$ with the finite set of vertices V that corresponds to the nodes in the network and the set of edges E which corresponds to the communication links between the nodes. If the network is *connected*, we mean that the underlying graph G is connected; i.e. between any two nodes there exists a (single-hop or multi-hop) communication path consisting of consecutive edges in G. A k-connected network implies that there are k independent paths among every pair of nodes [147]. For $k > 1$, the network can remain connected even when some of its nodes or links fail. In our work, our goal is to achieve at least 1-connected network.

3.1.1 Problem formulation

The self-configuration and self-healing problem can be formulated as:

Given N mobile nodes, how should they deploy themselves so that the resulting configuration maximizes the total coverage formulated in 3.1 with the constraint of keeping the network connected and how should they relocate to maintain coverage and connectivity whenever there is a failure.

Before looking at the algorithms which maximize and maintain area coverage with network connectivity, we look at two different area coverage problems: sensing range-based and communication range-based.

3.1.2 Sensing range-based coverage

In sensing range-based area coverage problems, we assume that all robots are equipped with some isotropic radial sensors of range r_s with which they sense and detect events in the environment. The area coverage in such cases can be calculated using equation 3.1, with $A_i = \pi r_s^2$; i.e. the coverage radius r is set equal to the sensing radius r_s.

The theoretically optimal coverage in terms of the number of nodes needed to achieve full coverage is when the nodes form a triangular lattice (or, equivalently hexagonal pattern with nodes at the center of each hexagon) [148] as shown in Figure 3.1.

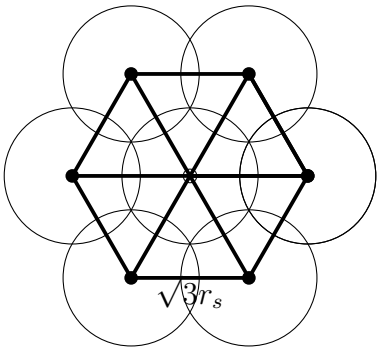


Figure 3.1: Optimal coverage

If we assume an isotropic radio communication of range r_c, within which the nodes can communicate with each other, following results are known in the literature [16]:

- When $r_c \geq 2r$ (with $r = r_s$ in this case), coverage of a region implies connectivity in the network.
- If $r_c \geq \sqrt{3}r$ (with $r = r_s$ in this case), then deploying sensors in the triangular lattice pattern shown in Figure 3.1 provides both coverage and connectivity, and is optimal in terms of number of sensors required.

As communication radius r_c is much greater than the sensing radius r_s in AMRoNet applications, a self-deployment algorithm that tries to achieve the optimal coverage configuration, automatically guarantees network connectivity.

3.1.3 Communication range-based coverage

In communication range-based area coverage problems, the coverage formulated in equation 3.1 is determined by setting the coverage radius r to the communication

range r_c. Hence the area covered by one robot A_i in this case is πr_c^2. Here also we assume an isotropic radio communication of range r_c for our robots.

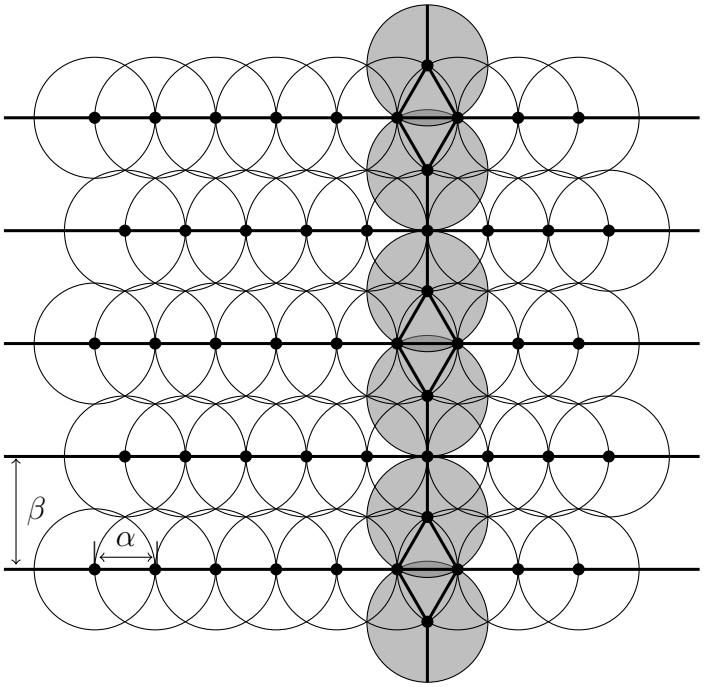


Figure 3.2: Optimal coverage when $r = r_c$

The theoretically optimal coverage and connectivity of the triangular lattice pattern shown in Figure 3.1 is valid only when $r_c \geq \sqrt{3}r$, but in communication range-based area coverage problems $r_c = r$. Hence, it is not the optimal solution. What is optimal in such cases is a strip based structure shown in Figure 3.2. It is asymptotically optimal for achieving both full coverage and 1-connectivity, not just for $r_c = r$, but for all $r_c < \sqrt{3}r$ values [16]. The strip can be constructed as follows:

- Nodes on the horizontal strips are placed on a line at regular intervals with a separation of $\alpha = \min\{r_c, \sqrt{3}r\}$.
- Horizontal strips are stacked with a vertical distance of $\beta = r + \sqrt{r^2 - \left(\frac{\alpha}{2}\right)^2}$ between the rows and alternate rows are shifted with an offset of $\frac{\alpha}{2}$.
- An additional vertical strip is placed in between the horizontal strips to connect the vertical strips.

Instead of placing one vertical strip, if two vertical strips are placed at the left and the right boundary of the deployment region, we get the optimal configuration that

guarantees coverage and 2-connectivity [16]. The optimality of this configuration to achieve full coverage and 2-connectivity is proved for all values of $\frac{r_c}{r}$, provided the vertical strips are removed when $r_c \geq \sqrt{3}r$ [16].

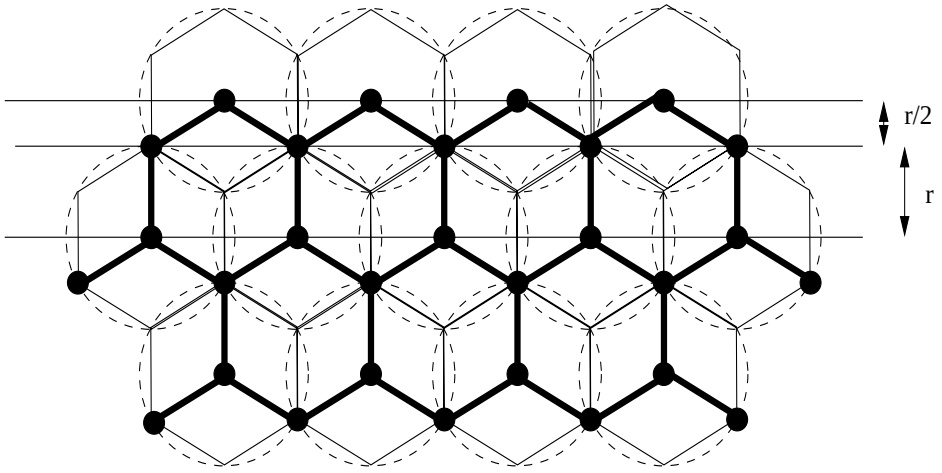
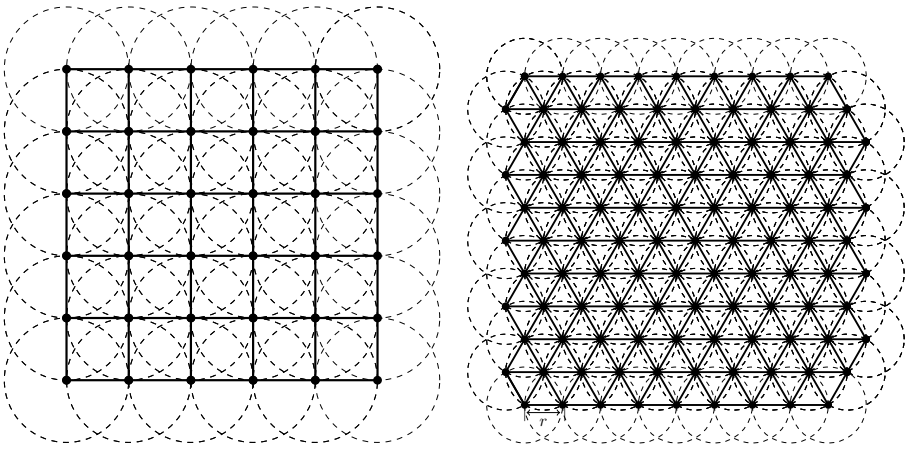


Figure 3.3: A hexagonal grid [94]

Efficiency of deployments can be measured by finding the *spatial density* which is defined as the number of nodes per unit area. The optimal strip-based deployment pattern has a density of $D_{STR} = \frac{0.536}{r^2}$, when $r = r_c$ [94]. Hexagonal grid structure shown in Figure 3.3 has a spatial density of $D_{HEX} = \frac{0.769}{r^2}$ [94]. It is often considered for practical deployment purposes, as it is more efficient than randomly constructed topologies [94]. Other commonly considered deployment patterns are square and triangular lattice structures shown in Figure 3.4a and Figure 3.4b respectively. Square grid has a spatial density of $D_{SQR} = \frac{1}{r^2}$ and triangular grid has a spatial density of $D_{TRI} = \frac{1.155}{r^2}$ [94].

3.2 Related Work

The coverage and connectivity problem has been studied previously in many related fields. The area coverage problem is related to the traditional art gallery problem [135] in computational geometry, where the objective is to find the minimum number of static guards to be placed in an environment, such that every point is monitored. The variant of the art gallery problem, Watchman Route problem [135], focuses on computing the routes the watchmen should take to guard an entire area. There exist different algorithms to solve these problems. However, all



(a) Square grid **(b)** Triangular grid

Figure 3.4: Regular deployment pattern

of them assume that a prior model of the environment is available.

In the static wireless sensor networks domain, proper node deployment is considered as a very important problem as it has a dramatic impact on the effectiveness of the network and the efficiency of its operation. Optimal node placement is a very challenging problem that has been proven to be NP-Hard for most of the formulations of sensor deployment and several heuristics have been proposed to find sub-optimal solutions [46, 49, 138]. Node placement strategies could be classified according to the deployment methodology into deterministic or stochastic strategies [92]. In deterministic strategies nodes are placed at predefined locations in a controlled manner whereas in stochastic strategies they are placed randomly. According to the optimization objective of the placement such as achieving maximal coverage, coverage with connectivity, network longevity, and data fidelity, several approaches has been presented [27, 43, 46, 68, 94, 148]. There are also role based placement strategies such as node acting as a sensor, relay, cluster head and base station [13, 25, 87, 124]. A comparison of the characteristics of the static node placement mechanisms is provided in [182].

There are similar problems considered in cooperative mobile robotics. The formation problem in which robots attempt to maintain a formation based on local sensing and computation [17, 56], Simultaneous Localization and Mapping (SLAM) [172] where robots aim to build a global map of the environment and the multi-robot exploration problem where they simultaneously explore different regions of an environment [28, 159] are good examples. Techniques such as frontier-based navigation [181], topological matching [45], fuzzy inference [125] and particle filters [171] are used for multi-robot exploration and map-building problems.

We are focusing on the self-deployment and self-relocation problems. Some researchers have explored the multi-robot dispersion problem which is quite similar to this problem. Although the main objective is to disperse robots into the environment, i.e. start out in a relatively compact space and spread out in an area, the term dispersion is used in a broad sense where it could also indicate multi-robot exploration problem. Some dispersion approaches focus on complete coverage of the environment with their sensors whereas others attempt to maximize their network coverage area. Hence a careful attention is needed to understand the correct objective of approaches listed as dispersion problem.

Let us now specifically look at the coverage and connectivity problems in multi-robotic and swarm robotic fields. A large body of literature is available in the domain of area coverage by multiple robots. A taxonomy of coverage algorithms is presented by Choset [38], which distinguishes the proposed approaches between offline algorithms, in which a map of the environment is known in advance, and online algorithms, in which map is unknown. The survey further divides the approaches for area coverage into two based on the methods they employ for decomposing the area: 1) approximate cellular decomposition where the free space is approximately covered by a grid of equally-shaped cells, and 2) exact decomposition where the free space is decomposed to a set of regions, whose union fills the entire area exactly. Another way of classification of existing approaches is: model-based approach where the environment is modeled or mapped during exploration and model-free approach which does not attempt to characterize the environment. Some algorithms are decentralized whereas others are centralized or semi-centralized. Some algorithms run completely parallel whereas some others run sequentially. We will now list some of the representative algorithms and classify them based on the approach they used for achieving coverage.

3.2.1 Force-based algorithms

The force based algorithms consider robots as virtual particles driven by virtual forces. The most popular force based algorithms use artificial potential field-based forces for coverage maximization. Potential field techniques for robotic applications were introduced by Khatib in [101] and have been widely used in the mobile robotics community for various tasks.

It has been first proposed in [89] for area coverage problem, where each robot is subjected to a force $F = -\Delta U$ which is the gradient of the scalar potential

U. Obstacles and other robots exerts a repulsive force F_{cover}, which is inversely proportional to the square of the distance between them. A viscous frictional force makes the system attain a state of static equilibrium. Later in [147], an extension of this approach to assure K-connectivity has been presented, where an attractive force $F_{connect}$, refrains the node degree getting too low by making them attract when the node degree becomes critical ($\leq K$). The repulsive force F_{cover} and attractive force $F_{connect}$ exerted on node i by its j^{th} neighbor are:

$$F_{cover}(i,j) = \left(\frac{-K_{cover}}{distance_{ij}^2}\right) \underbrace{\left(\frac{p_i - p_j}{|p_i - p_j|}\right)}_{unitvector} \tag{3.2}$$

$$F_{connect}(i,j) = \begin{cases} \left(\frac{K_{degree}}{(distance_{ij} - r_c)^2}\right)\left(\frac{p_i - p_j}{|p_i - p_j|}\right), \text{ if critical connection;} \\ 0 \quad \text{otherwise.} \end{cases} \tag{3.3}$$

where p_i and p_j are the position of nodes i and j respectively, and K_{cover} and K_{degree} are the force constants. We refer this approach as $Force$ in our article.

A virtual force algorithm to enhance the coverage of an initial randomly deployed sensors has been presented in [187], where the net force on a node is sum of three forces a) repulsive force exerted by obstacles, b) attractive force exerted by areas of preferential coverage and c) an attractive or repulsive force by another node depending on its distance and orientation. Once the effective sensor positions are identified, a one-time movement is carried out to redeploy the sensors at these positions.

Inspired by the equilibrium of molecules, a Distributed Self Spreading Algorithm ($DSSA$) has been presented in [82], where the force exerted on node i by its j^{th} neighbor is calculated as:

$$F(i,j) = \frac{D}{\mu^2}(r_c - \underbrace{|p_i - p_j|}_{distance}) \underbrace{\frac{p_j - p_i}{|p_j - p_i|}}_{unitvector}$$

where D is the current local density and μ^2 is the expected average density which is $\frac{N \cdot \pi \cdot r_s^2}{A}$.

3.2.2 Incremental self-deployment algorithms

Another self-deployment approach is the incremental greedy deployment algorithm where nodes are deployed sequentially by making use of the information of previ-

ously deployed nodes. A robot carrying multiple immobile nodes and deploying them in an unexplored area with the assistance of previously deployed node has been presented in [21]. In [88] and [90], an incremental and greedy self-deployment algorithm for mobile nodes is presented which is similar to the frontier-based approach [181]; however, here the occupancy maps are built from live sensory data. The goal is to maximize network coverage under the constraint that nodes maintain line-of-sight with each other. One obvious drawback of the algorithm would be, it is very difficult for the network to reconfigure itself, which is one of the main advantages of using mobile robots.

3.2.3 Spanning tree-based algorithms

Spanning tree-based approach is also used to solve coverage problems in known and unknown environments. In this technique each robot decomposes its environment into cells of equal size referred to as approximate cellular decomposition by Choset [38] and uses a minimum spanning tree-based algorithm to determine the path to traverse while navigating those cells. Unlike other area coverage or exploration problems where a sensory coverage of an unknown environment is sufficient; here the focus is on the physical sweeping of a tool over every point of a given work-area. A polynomial time Spanning Tree Coverage algorithm (STC) for complete offline and online coverage of the terrain by a single robot is introduced in [66].

Hazon et al. extended the idea of STC in [78] to a multi-robot setting where the problem is to compute the trajectory for each robot so that the cover time (the largest travel cost of any robot) is minimized. This problem is NP-complete. The approach called as Multi-Robot Spanning Tree Coverage (MSTC) described in two versions: a non-backtracking MSTC, and backtracking MSTC, guarantee robust, time-efficient and complete coverage. A polynomial-time multi-robot coverage heuristic, Multi-Robot Forest Coverage (MFC), based on an algorithm for finding a tree cover with one tree for each robot with balanced weights (travel costs) is presented in [184]. In [79], an online version of STC coverage that runs in parallel and builds a local spanning tree of uncovered cells by a depth-first-like procedure which generates the path for the controlled robot is proposed. For the boundary coverage problem of regular structures, a minimal spanning-tree-based algorithm to achieve complete coverage of an unknown grid by incrementally constructing the tree and using a low-level reactive control with deliberative planning for traversing the tree is presented in [41]. The approaches in [184] and [41] assume

that the environment is known a priori, either fully or partially, and its graphical representation is available to the robots.

3.2.4 Voronoi-based algorithms

Voronoi-based algorithms use the structure of Voronoi diagrams for finding coverage holes and minimizes them by relocating the robots. In [177], three Voronoi based algorithms VEC, VOR and Minimax, has been proposed. VEC uses Voronoi polygon for finding coverage holes, but its movement is based on virtual force exerted by neighbouring sensor and field boundary. In VOR, if several coverage holes are detected, the node would move towards its farthest Voronoi vertex to cover the hole; whereas in Minimax the node moves to a point called Minimax point which reduces the variance of the distances to all the Voronoi vertices of the Voronoi polygon.

In [178], a bidding protocol for hybrid sensor networks where the static nodes find coverage holes by constructing Voronoi diagram and bid the mobile nodes to move to the holes is introduced.

3.2.5 Pheromone-based algorithms

Ants and other insects use chemical substance called pheromones for various communication and coordination tasks [176]. An ant inspired heuristic for distributed area coverage is presented in [176]. Here robots can leave chemical odor traces that evaporate with time, and are able to evaluate the strength of smell. The environment is decomposed into cells (grids) and on visiting a cell robots deposit the virtual pheromones. Robots use either a hill-climbing approach called ANT-WALK-1 or a multi-level depth first search with backtracking called ANT-WALK-2 to select the next cell to visit. A similar grid-based approach for cell decomposition and an ant inspired mechanism for selecting the next action is presented in [105]. They also assume that robots can leave behind physical trails within the environment which can be sensed by other robots or all robots deposit virtual pheromone on a centralized pheromone map that is shared by all robots. The coverage information is exchanged through the pheromone trails. In [137], each robot deposits virtual pheromone within a map of the environment (or part of the environment) maintained within its memory. At certain intervals robots communicate their local virtual pheromone maps to each other and fuse the virtual pheromone obtained from other robots.

The virtual pheromone concepts applied to the problem of dispersion is introduced in [140]. A virtual pheromone is encoded as a single modulated message consisting of a type field, a hop count field, and a data field. Each robot sends the message to its neighbors and they forward it by decreasing the hop count till the hop count reaches zero. Similar to the potential field approach robots attract and repel each other based on the strength of received virtual pheromone messages which in turn depends on the distance of neighboring robots. They also use the concept of *bud* where one robot designated as bud gets a repulsive force stronger than attractive force, which pushes it further away and other robots then expand to fill in the space to maintain connectivity.

In [141], a dispersion behavior for a group of miniature robots inspired by insect colony coordination behavior is introduced. It uses *repellent virtual pheromones* to guide several miniature robots (Scouts) to get deployed and dispersed quickly. An overhead camera or a camera mounted on a command/control robot provides each miniature robot the estimation of positions of other nearby robots, using a vision-based analysis of the location of the colored markers on these robots. Virtual pheromones are modeled similar to potential fields and cumulative repellent force of virtual pheromones is calculated to find the moving direction.

Another pheromone-based algorithm called Mark-Ant-Walk for coverage maximization is presented in [136]. In this algorithm the ant-like robot marks a radius around them and then moves to the point with the least marking within visual range. It requires accurate pheromone marking and sensing the level of the pheromone marks. Sensing pheromone marks of other robots provides an indirect communication between the robots. The algorithm provides a theoretical bound on covering time, and ensures complete and efficient covering of arbitrary connected domains.

3.2.6 Behaviour-based algorithms

One of the earlier works in this direction is the self-organizing behavior in swarm robots that search for pollutants [71]. A behavior-based approach for dispersion of robot-teams by using a random-wandering behavior coupled with moderate robot repulsion as well as more significant obstacle repulsion is addressed in [15]. Inspired by the group behavior seen in nature, combining basic behaviors with higher level group behaviors such as aggregation, and dispersion can be synthesized [127]. In the synthesized dispersion behavior, agents move away from the centroid of the

local density distribution of the other agents that are visible to their sensors.

In [180], autonomous robot dispersion for mobile nodes in a scenario where mobility is required to cover the entire region due to a lack of wireless network connectivity has been presented. The author used a random diffusion method for node deployment while collecting data over a fixed surveillance region. The behavior-based approach focuses on developing a set of simple local behaviors for maximizing area coverage.

In [22], four behaviors are specified: *obstacle avoidance*, *walk*, *observe*, and *dance*. Obstacle Avoidance causes robots to steer away from each other and other objects in the environment. Walk causes a robot to move forward in the direction it is currently facing. Observe behavior chooses the most promising direction for exploration. Dance behavior is implemented differently for two spreading techniques: Informative and Molecular. In the Informative technique the dancer robot performs a stylized motion which is observed by the observer robot. A local coalition is formed to decide the subsequent motion that increases total coverage based on the exchanged relative position and bearing information. In the Molecular Technique, it relies only on vision and the dancer selects a direction which is diametrically opposite to the average angle subtended by all its neighbors in its visual field. It is thus repelled away from its neighbors.

Dispersion of mobile robot swarms without any active communication is presented in [130]. Four different simplistic coverage strategies namely, a *random walk* behavior in which robots move forward with a small random turn factor updated at short intervals, a *follow wall* behavior to find obstacles to follow and thereby explore an environment, a *seek open* behavior to move in the opposite direction of the average obstacle vector and a *fiducial* behavior in which a fiducial device allowed robots to localize relative to each other and use that information to move away from each other, are proposed in the paper. Robots are assumed to have line of sight communication to avoid collision with each other and obstacles. Experimental results with simulated robots in different environments show that fiducial behavior performed is the best.

In [128], a Directed Dispersion algorithm is presented which has a *disperse uniformly* algorithm to spread robots evenly and a *frontier guided dispersion* algorithm to direct robots towards unexplored areas. Another frontier guided dispersion algorithm is described in [91]. Robots use a wall-following algorithm with one robot acting as the leader who detects the frontier from its neighboring cells. If the leader fails to detect the frontier, strategies such as breadth-first and depth-

first search are used to determine a frontier. In [151], a utility based frontier guided mechanism where each robot selects between maintaining connectivity with the rest of the team and visiting frontier cells is presented.

In [137] a semi-centralized algorithm that combines artificial potential fields and behaviors together with leader election, counting hops from the leader, and sending alarms to prevent disconnections, has been presented. The artificial potential fields use three forces: The first one is a *radial force* whose magnitude is determined by distance to obstacles. The second one is an *open force* that pulls a robot towards empty regions and the last one a *tangential force* to make the robot follow the wall. The robots select a leader and use the leader to coordinate movements. The leader does not move and provides an anchor for the entire network. Robots count hops to the leader and when they move, at least one neighbor with a lower hop count is maintained. To prevent disconnections alarm messages are sent. It uses the following set of behaviors: *random, random turn, scatter, forward, backward* and *freeze* for dispersion.

3.2.7 Wireless signal intensity-based algorithms

Dispersion of a swarm of robots based on wireless signal intensity has been addressed in [126,174]. Even though they do not know the relative locations of other robots, using the wireless signal intensity information of neighbors, the direction to maximize coverage has been determined. In [126], the wireless signal intensity is assumed to be inversely proportional to the square of the distance between the robots. In the *Clique Intensity Algorithm* they proposed, one robot is selected as *sentry* (stationary) for every maximal clique in the connectivity graph. Other robots in the clique attempt to move away from the sentry by monitoring the change in the signal intensity over time. Robots share connectivity information with each other so that they can all agree on a set of sentries by following a set of specific rules.

The signal intensities are realistically modeled using sampling technique, taking both the distance and relative orientations of the wireless sensors into account in [174]. If the wireless intensity decreases during their movement (for same orientations), it can be deduced that the robot is moving away from other robots. An inversely proportional relation between wireless signal intensity and distance between robots would hold for small changes in relative orientations. Based on this assumption, moving direction for dispersion is found out.

3.3 Our design direction

We are focusing on the self-deployment aspect of nodes with an objective of maximizing area coverage and the self-relocation aspect which maintains network connectivity in case of failures. The robot deployment and relocation are then completely autonomous processes. In our scenarios prior models of the environment are incomplete, inaccurate or non-existent. We also concentrate on model-free approaches which do not attempt to characterize the environment, as model based approaches which create maps or other internal representation of the environment are expensive in terms of storage and computation requirements. This is particularly important when we consider small and low cost robots such as swarm robots as AMRoNet nodes. Moreover, the approaches that do not keep track of the past actions and coverage history of all robots in the environment or itself, are particularly interesting for such swarm robotic AMRoNets with limited capabilities.

In our design, we focus on fully distributed and localized approaches where each node need to gain only limited knowledge of the environment. Hence centralized, semi-centralized approaches are not suitable for our purpose. In localized approaches, robots need not know how many other robots are there in the environment, where they are located or where those robots have been. Thus expensive storage and computation requirements are completely avoided.

We plan to adopt swarm-based techniques for our algorithmic design, which are indeed based on local rules. The global behavior of the system such as achieving coverage and connectivity emerges from the local interactions based on these simple rules. The main challenge in the swarm-based design is to identify correct local rules that lead to such emergent behaviors. In the next chapter, we introduce such an algorithm for coverage maximization with connectivity

CHAPTER 4

Sensing Range-based Coverage

By viewing Nature, Nature's handmaid Art,
Makes mighty things from small beginnings grow.

John Dryden

In this chapter we focus on applications that need *sensing range-based coverage* with *network connectivity*. We design an algorithm that achieves both in practical scenarios. We focus on using bio-inspired algorithms in our design, as they work well in environments where prior-knowledge about the environment is minimal and adapt to unforeseen changes in the task environment quickly. The key idea of these algorithms is to write simple local rules to achieve coverage and connectivity as an emergent property of the algorithm.

The main idea used in our swarm-based coverage and connectivity maintenance algorithm is born from the schooling behavior of fish which "optimize" the coverage-connectivity $C - C$ constraint naturally, i.e.

- The swarm needs to stay together to appear as one fish. So they maintain connectivity.
- The appearance needs to be as large as possible to frighten predators. So they try to increase the coverage.

Thus, nature offers a natural concept for solving the coverage-connectivity problem. We look at this natural model and design our new swarm-based coverage and connectivity maintenance algorithm. We show that our bio-inspired approach achieves better and faster coverage.

4.1 Preliminaries

Let N denote the total number of robots and R denote an individual robot. The environment where the robots spread out is a 2-D area, denoted as A. No prior map of the environment is available.

We assume that our robots have isotropic radial sensors of range r_s, with which they sense and detect events in the environment and an isotropic radio communication of range r_c, with which they can communicate with other robots. The radio communication range is usually very large compared to the sensing range. When $r_c \geq 2r_s$, coverage of a region implies connectivity in the network [16]. Hence we set $r_c \geq 2r_s$ in our experiments.

An important objective of our algorithm is to *maximize the total area covered* with network connectivity. A metric commonly used in evaluating the performance of $C-C$ algorithms is *rate of coverage*, which indicates the increase or decrease in coverage over time. Our next important objective is to design an algorithm that has good *rate of coverage*. A third important objective is to design an algorithm which *maintains coverage over time*. This means that whenever a coverage hole appears, due to failures of robots or changes in the environment, after all robots have reached their equilibrium state, they should be able to recover it by relocating the robots.

4.2 Swarm-based Algorithm

4.2.1 Swarming rules

The swarm-based coverage and connectivity maintenance algorithm is inspired from the schooling behavior of fish where fish usually of the same species, age and size stay together and maximize "coverage". The basics of schooling behavior could be summarized in 3 rules given below [165]:

- Avoid collisions with your neighbors: In the zone of repulsion, the focal fish seeks to distance itself from its neighbors in order to avoid a collision.
- Remain close to your neighbors: In the zone of attraction, the focal fish seeks to move towards a neighbor.
- Move in the same direction as your neighbor: In the zone of alignment, a focal fish seeks to align its direction of motion with its neighbors.

Basic rules behind the schooling behavior are same in all animal aggregation behaviors such as flocking of birds, the swarming of insects, and herding of land animals. Reynolds [150] first modeled the flocking behavior with his simulation program called Boids and uses the following three rules in order of decreasing precedence to simulate flocking.

- Collision Avoidance: Avoid collisions with nearby flockmates.
- Velocity Matching: Attempt to match velocity with nearby flockmates.
- Flock Centering: Attempt to stay close to nearby flockmates.

4.2.2 Mathematical modeling of schooling behavior

To solve the $C-C$ problem, we use a force-based variant of these rules instead of the priority-based behavioral model used in Boids [150]. The force-based variant of the swarm rules are:

- $F_{separation}$**:** A force which pushes away from neighbors, increasing the size of the swarm.
- $F_{cohesion}$**:** An attractive force towards the centroid of the neighbors, maintaining the connectivity of the swarm.
- $F_{alignment}$**:** A force matching the average force of the neighborhood.

The three forces $F_{separation}$, $F_{cohesion}$ and $F_{alignment}$ exerted on each member of the swarm are calculated as follows:

$$F_{separation} = \frac{1}{|neighbors|} \sum_{i \in neighbors} \frac{K}{(|\vec{p} - \vec{p_i}|)^2} \frac{\vec{p} - \vec{p_i}}{|\vec{p} - \vec{p_i}|} \tag{4.1}$$

$$F_{cohesion} = \left(\frac{1}{|neighbors|} \sum_{i \in neighbors} \vec{p_i} \right) - \vec{p} \tag{4.2}$$

$$F_{alignment} = \frac{1}{|neighbors|} \sum_{i \in neighbors} \vec{F_i} \tag{4.3}$$

where p is the position of the current member of the swarm, $\vec{p_i}$ is the position of its i^{th} neighbor, $\vec{F_i}$ is the force of its i^{th} neighbor and K is a *repulsion* parameter.

The resulting force F on each robot is the weighted sum of the three forces $F_{separation}$, $F_{cohesion}$ and $F_{alignment}$

$$F = w_1 \cdot F_{cohesion} + w_2 \cdot F_{separation} + w_3 \cdot F_{alignment} \tag{4.4}$$

where w_1, w_2 and w_3 denote the weights of the forces. The sum of the weights w_1, w_2 and w_3 is set to 1. Usually the weight w_1 is set equal to w_2 and w_3 to a smaller value compared to w_1 and w_2. Setting w_3 to a small value ensures that the alignment force gets weakened each time it spreads through the network and results in a more stable behavior. It is also possible to set the alignment weight to zero.

A robot reaches a state of equilibrium when the total force exerted on it is zero. We design the repulsion parameter K in such a way that when a robot reaches an aimed distance D_{AIM} from its neighbors, the sum of two forces, $F_{cohesion}$ and $F_{separation}$ becomes zero. Let n be the number of neighbors of the current robot. At the aimed distance,

$$F_{cohesion} + F_{separation} = 0 \tag{4.5}$$

$$\begin{aligned} F_{cohesion} &= (\frac{1}{n}\sum_{i\in n}\vec{p_i}) - \vec{p} \\ &= \frac{1}{n}(\vec{p_1} + \vec{p_2} + \ldots + \vec{p_n}) - \vec{p} \\ &= \frac{\vec{p_1} + \vec{p_2} + \ldots + \vec{p_n} - n\vec{p}}{n} \end{aligned} \tag{4.6}$$

$$\begin{aligned} F_{separation} &= \frac{1}{n}\sum_{i\in n}\frac{K}{(|\vec{p}-\vec{p_i}|)^2}\frac{\vec{p}-\vec{p_i}}{|\vec{p}-\vec{p_i}|} \\ &= \frac{1}{n}\sum_{i\in n}\frac{K\cdot(\vec{p}-\vec{p_i})}{(|\vec{p}-\vec{p_i}|)^3} \\ &= \frac{1}{n}\frac{K((\vec{p}-\vec{p_1}) + (\vec{p}-\vec{p_2}) + \ldots + (\vec{p}-\vec{p_n}))}{(|D_{AIM}|)^3} \\ &= \frac{-1}{n}\frac{K(\vec{p_1} + \vec{p_2} + \ldots + \vec{p_n} - n\vec{p})}{(|D_{AIM}|)^3} \end{aligned} \tag{4.7}$$

From equation 4.6 and 4.7, we calculate K as follows:

$$\begin{aligned} \frac{\vec{p_1} + \vec{p_2} + \ldots + \vec{p_n} - n\vec{p}}{n} &= \frac{1}{n}\frac{K(\vec{p_1} + \vec{p_2} + \ldots + \vec{p_n} - n\vec{p})}{(|D_{AIM}|)^3} \\ K &= (|D_{AIM}|)^3 \end{aligned} \tag{4.8}$$

When the number of neighbors of a robot drops to zero, due to failure of all its neighbors or other errors in the system, the robot starts a *random walk* behavior to find a possible new neighbor.

4.2.3 Neighborhood definition

There are two basic models used to define the neighborhood of bird flocks or fish schools [18]:

- **Topological model:** Nearest n members of the swarm regardless of distance.
- **Metric model:** All members of the swarm within certain fixed range r_n.

In the topological model the number of interacting individuals is fixed irrespective of their metric distance. There are always at most N neighbors and there is no difference between the force exerted by a neighbor who is far away or near. However, in the metric model, the number of neighbors varies with density, i.e., a fixed range metric model has more neighbors in a very dense swarm than a very sparse one. Moreover, the neighbors which are nearer are more important (exert more force) than those which are far away.

In the formalization of the forces given in equations 4.1, 4.2, and 4.3, we also use the metric model, where the impact of force is weighted according to the metric distance. However, recent empirical results [18] show that swarms use topological model with n about $6 - 7$. A pure topological model is not practical in swarm robotics due to the limitation of finding n neighbors irrespective of the distance. Hence, we propose a hybrid topological model for the swarm robots, where we take at most n nearest neighbors in the force calculation instead of all neighbors within the fixed range r_n.

4.2.4 Obstacle avoidance

In realistic scenarios robots always encounter obstacles. So they need an obstacle avoidance algorithm that fits with the force calculation. Each robot has a sensor or sensor array that detects the obstacles. When they detect an obstacle a force $F_{obstacle}$ is calculated based on the direction and the distance of the obstacle.

The force is calculated in the same way as $F_{seperation}$ is calculated, but the magnitude of the force is changed by multiplying with a factor $1/distance$, i.e.

force from an obstacle is of order $O(\frac{1}{distance^3})$. This is because of the following reason: If a robot detects an obstacle which is getting pushed by many neighboring robots from the other side, then $F_{seperation}$ adds up too much compared to $F_{obstacle}$ due to the outdated position information of the robots. This is inevitable due to the delay in the sense-act loop in estimating the robots' actual positions. This causes the robot to crash into the obstacle. In order to avoid deadlock situations where the total force becomes zero, a Gaussian noise with 0-mean and a small standard deviation of 0.02 arbitrarily selected is added to the sensor values during the $F_{obstacle}$ calculation.

4.3 Experimental Analysis

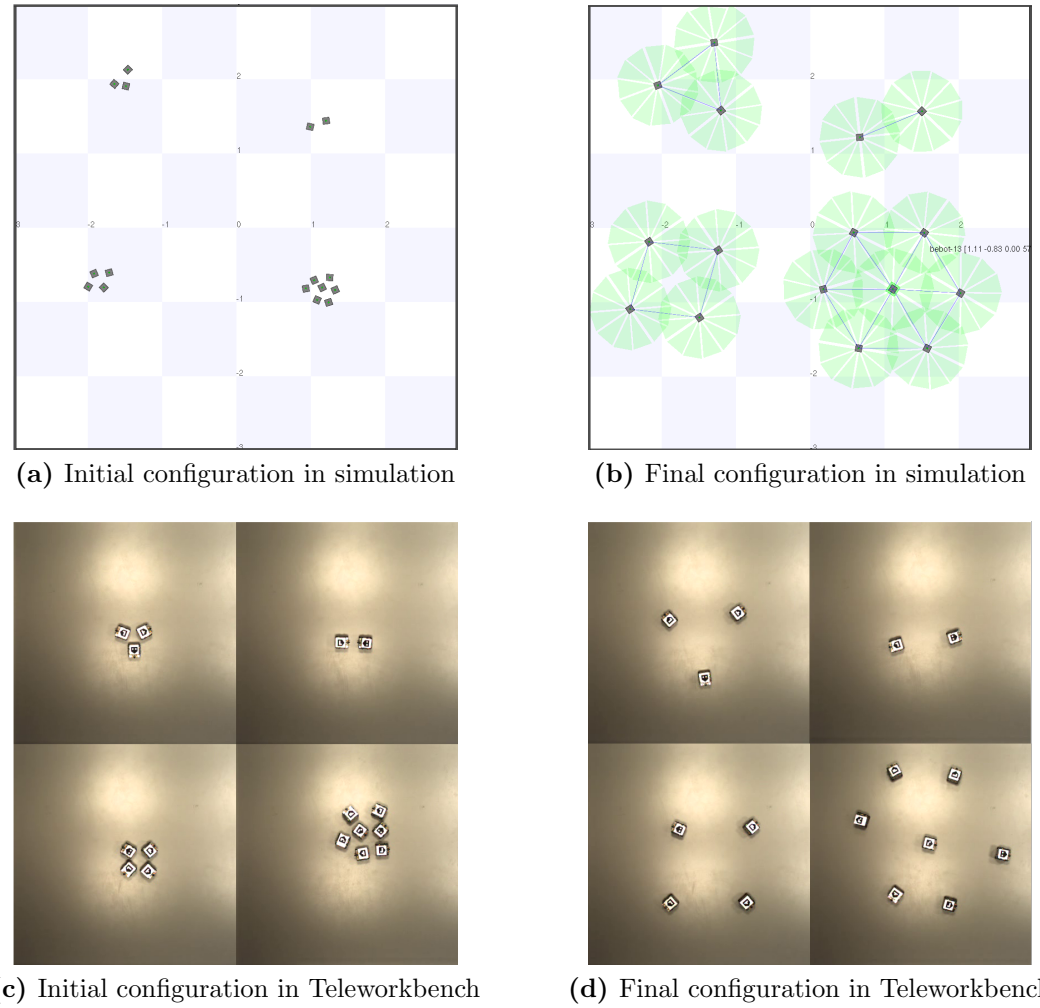
We use the Player-Stage robotic platform and Bebot robots for our experiments. Experiments with the Bebot robots are conducted in the Teleworkbench environment. The Teleworkbench server estimates the position of the robots and writes them on to a *Blackboard* device provided by Player, whenever there is a position update. Each robot communicates with the *Blackboard* proxy and estimates the position of its neighbors. Simulation based experiments use a virtual model of the Bebot robot with infrared range sensors and wireless units. The position of neighbors is easily obtained from the simulator.

In our experiments each robot tries to spread up to D_{AIM} from each other based on the equation 4.4, with weights w_1 and w_2 set to 0.4 and w_3 set to 0.2. Coverage increases with the distance of separation between two robots until it reaches $2r_s$. An optimal coverage in terms of number of robots needed to cover an area is achieved when they form a triangular lattice pattern with inter-robot separation of $\sqrt{3}r_s$. Hence, we set D_{AIM} to $\sqrt{3}r_s$ with an r_s value of 0.5 m for simulation based experiments. In real experiments, as Bebot robots have infrared sensors of maximum range 20 cm, we set r_s to this value. When $r_c \geq 2r_s$, then coverage of region implies connectivity in the network and hence we set r_c to the minimum value $2r_s$ for our experiments. The fixed range r_n used in the definition of neighborhood is set to r_c; i.e. each robot considers all robots located within a distance of r_c as its neighbors.

4.3.1 Geometric patterns

In this experiment each robot tries to spread up to the D_{AIM} from each other and forms the geometric patterns such as line, triangle, square and hexagon (triangular

lattice) when the number of robots is varied between 2, 3, 4 and 7. We perform this experiment with the Stage simulator and with Bebot robots in the Teleworkbench environment.



(a) Initial configuration in simulation

(b) Final configuration in simulation

(c) Initial configuration in Teleworkbench

(d) Final configuration in Teleworkbench

Figure 4.1: Geometric patterns using swarm-based algorithm

Figure 4.1a and Figure 4.1c shows the initial configuration in the Stage and Teleworkbench environment respectively. Despite the unsynchronized execution of the algorithm in simulation environment, the robots reach the final configuration as shown in Figure 4.1b and create almost perfect geometric patterns. In Teleworkbench environment, the robots are completely independent and many additional issues arise, e.g. delay in communication, imprecise location estimation etc. Still they are able to reach the final configuration as shown in Figure 4.1d and create almost perfect geometric patterns.

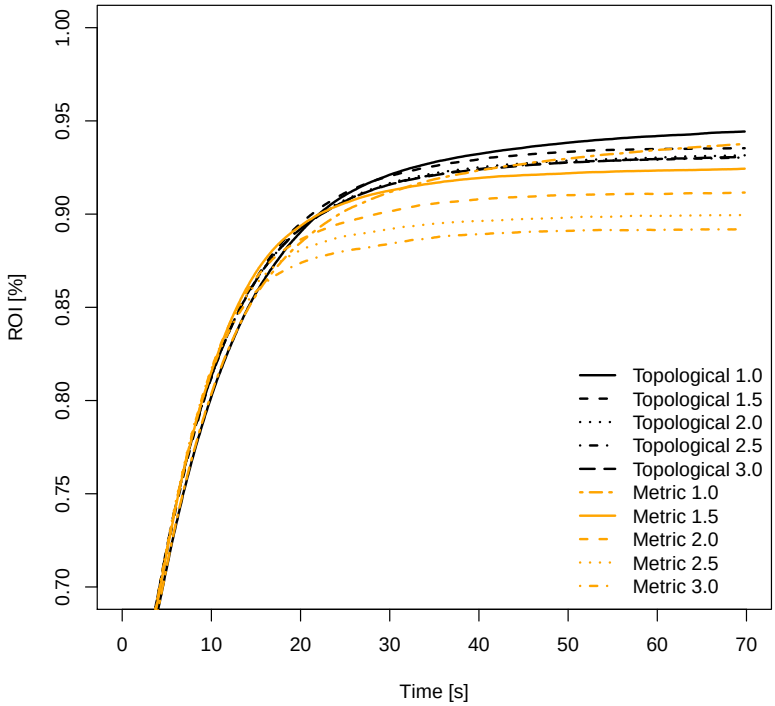


Figure 4.2: Comparison of coverage in metric and topological neighborhood model

4.3.2 Topological and Metric model

In this experiment we test the effect of neighborhood models on coverage using a simulation based analysis. As explained in Section 4.2.3, the metric model considers all members within certain fixed range as its neighbors, whereas the topological (hybrid) model considers at most 6 nearest neighbors within this fixed range. We set the minimum fixed range r_n to $2r_s = 1.0$, to keep the robots connected even at maximum possible coverage.

We consider an empty field of size 6×6 meters and deploy 56 robots randomly in the field. This is the minimum number of robots needed to achieve full coverage based on the triangular lattice structure. As the number of neighbors varies with density in metric and hybrid topological models, to test its effect on coverage, we vary r_n and record the coverage over time. Each experiment is repeated 50 times with different initial configurations.

Figure 4.2 shows the results of these experiments when r_n is varied from 1.0 m to 3.0 m at steps of 0.5 m. The curves are averaged over the 50 independent runs and show the covered region of interest (ROI) at a certain point of time. They indicate that coverage increases over time rapidly during the initial phase and becomes constant towards the end. They also show that the hybrid topological

model is better than the metric model in coverage and stability. The metric model has more variation with density changes, whereas the hybrid topological model is not much affected by the change in density.

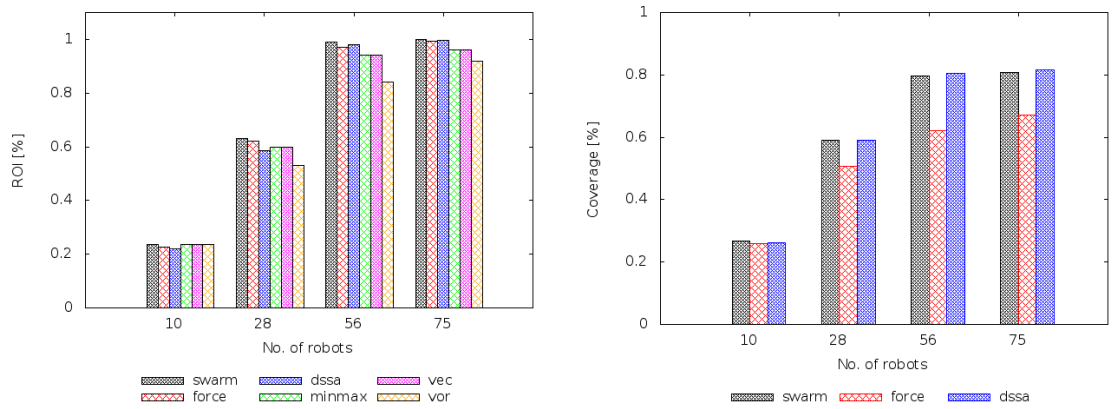
4.3.3 Comparison of performance

We now compare the performance of our swarm-based algorithm, referred to as $Swarm$, with the state of the art algorithms. From Section 4.3.2, we find that the topological (hybrid) neighborhood model is better than the metric model in coverage and stability. So here onwards we consider only this model in our experiments, with the fixed range set to the minimum value, i.e. $r_n = 2r_s$.

Among the state of the art approaches mentioned in Section 5.2, our approach matches best with the force-based approaches and we use them as our references for comparison. Voronoi-based algorithms described in Section 5.2 also use similar assumptions, but need to be extended to make them work in our scenarios. We consider them in those experiments where these extensions are possible. Other approaches mentioned in Section 5.2 have different objective and use different assumptions. Hence, we do not consider them in our analysis.

For the $Swarm$ algorithm, we find the net force on each robot according to equation 4.4. The implementations of $Force$ and $DSSA$ algorithms are straight forward. For $Force$ implementation, we find the constants K_{cover} and K_{degree} of equation 3.2 and 3.3 exactly as mentioned in [147] and assign the same values 0.25 and 0.8 for the damping and safety factors used in the paper. For $DSSA$, the expected density μ and the local density D are calculated exactly as mentioned in [82].

During the implementation of Voronoi based algorithms, the following issue arises: If robots are cluttered together in a small area, the robots at the border of the cluttered area might construct open Voronoi polygons. Checking the existing Voronoi points cannot detect coverage holes in such cases and the coverage is unimproved. To solve this issue, we assume that the area where the robots are deployed is defined with known boundaries. When the Voronoi polygons are open or the existing Voronoi points are located outside the boundary, new Voronoi points from the boundary are added. This makes the polygons closed within the defined area and moves the robots from cluttered regions towards sparser regions.



(a) Without obstacles **(b)** With obstacles

Figure 4.3: Comparison of coverage performance on a square map

4.3.3.1 Experiment 1 - Coverage analysis:

The first experiment to compare the performance of the algorithms is conducted on a square field of size $6 \times 6\ m^2$ for the simulation based analysis and $2.4 \times 2.4 m^2$ for the Teleworkbench based analysis. In the latter case, we deploy 10 Bebot robots randomly in the given area, whereas in the former case, we consider three scenarios, an under-deployed with 28 robots, an exact deployed with 56 robots and an over-deployed with 80 robots. During these deployments, we consider different initial configurations such as robots deployed in one small region at the center of the field or at two corners of the field or all corners of the field or random deployment in the entire field. We consider 50 different configurations for simulation-based experiments and 5 for real robot experiments.

We also consider scenarios with different obstacles in the field, e.g. Figure 2.3. Force from the obstacles $F_{obstacle}$ specified in Section 4.2.4 is added to the force calculated by the $Swarm$, $DSSA$ and $Force$ algorithms. However, such an addition does not suit the Voronoi-based algorithm as it is not force-based and we do not consider them in scenarios with obstacles.

Figure 4.3a shows the performance of $Swarm$, $DSSA$, $Force$, VEC, VOR and $Minimax$ algorithms averaged over the different independent runs in scenarios without obstacles. It shows that the $Swarm$ algorithm (in black color) achieves better coverage than the state of the art algorithms. The new force-based rules allow them to reach the optimal triangular lattice structure better than the state of the art algorithms.

$Minimax$ algorithm performs better than VOR and VEC algorithms in [177].

However, in our experiments VEC achieves better coverage than $Minimax$ and VEC. One reason for this variation could be the difference in the simulator, where the robots do not move to the Minimax point or farthest Voronoi vertex in each iteration of the algorithm due to the randomness and unsynchronized execution of the algorithms in the Stage simulator.

Figure 4.3b shows the performance of $Swarm$, $DSSA$ and $Force$ algorithms in scenarios with obstacles. In these experiments, ROI is the total area minus the area occupied by the obstacles. Figure 4.3b shows that the performance of our algorithm is better than the $Force$ algorithm and same as $DSSA$ algorithm.

4.3.3.2 Experiment 2 - Coverage maintenance

In this experiment we test how fast the algorithm regains coverage in case of failure of robots or coverage holes using simulation based analysis. For this, we consider an over-deployed scenario with 80 robots in a square field of size 6×6 meters without obstacles. Robots are initially deployed randomly in the given area and are allowed to spread in the area as in Experiment 1. Now we simulate a coverage hole or failure of robots by removing all robots located within a distance of 2 m from the center point of the field.

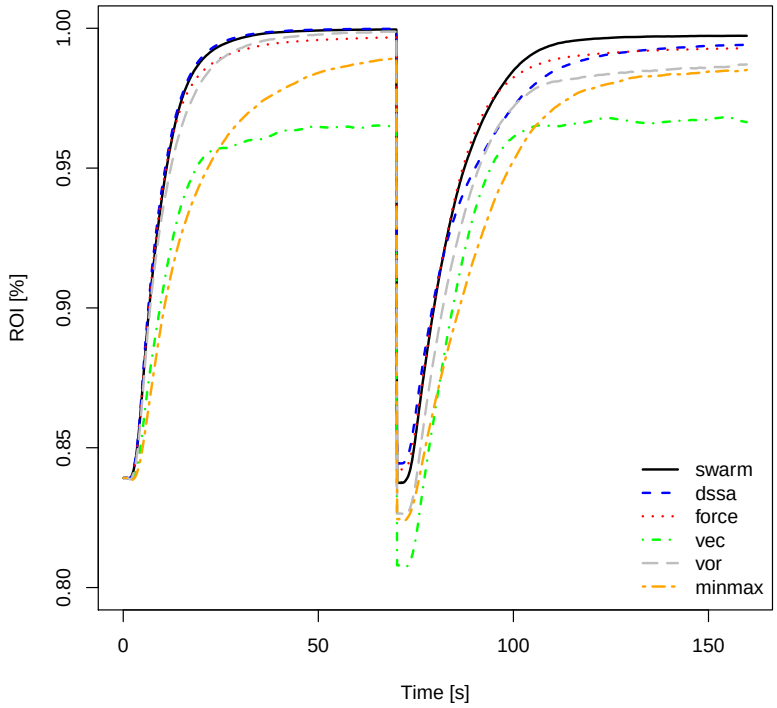


Figure 4.4: Comparison of coverage performance on relocation scenario

Figure 4.4 shows how fast and good the robots relocate according to the respec-

tive algorithm, to regain coverage. It shows that the *Swarm* algorithm performs better than the state of the art algorithms by recovering from failure better and faster. The faster coverage regain of the *Swarm* algorithm is mainly achieved by the alignment rule, which uses the information about the failure much better than other force based algorithms. In differential drive robots where robots have to turn to change their direction, alignment of direction is useful. As we are not just aligning the direction, but averaging the neighbors' force, the robot considers alignment force as a forecast of what its neighbors are performing and starts adequate counter actions earlier. These factors contribute for the improved performance of our algorithm.

Failure of a large number of robots may cause network partition. However, in our scenarios we assume that there are more redundant robots than the number of robots failing. Hence, even when robots fail and network may get partitioned, as the total number of working robots in the system is always greater than the optimal number of robots needed for full coverage, the network will get connected after relocation.

4.4 Summary

In this chapter, we proposed a new swarm-based coverage and connectivity maintenance algorithm inspired from the schooling behavior of fish. The algorithm is a force-based variant of the local rules seen in such animal aggregation behaviors. The three basic rules namely separation, cohesion and alignment are formalized into equivalent forces. The separation force pushes the robots away from their neighbors and increases the size of the swarm. The cohesion force maintains the connectivity of the swarm. The alignment force keeps the robots aligned to their neighbors and makes relocation faster. Empirical analysis shows that our bio-inspired swarm-based algorithm achieves better and faster coverage than the state of the art algorithms tested, in scenarios with and without obstacles. Moreover, it regains coverage faster than these algorithms from the coverage holes occurring due to robot failures.

CHAPTER 5

Communication Range-based Coverage

In this chapter we focus on AMRoNet applications where the nodes act as a temporary infrastructure to facilitate communication between *agents*. The agents could be humans or powerful robots moving autonomously on a terrain for missions such as urban search and rescue or exploration of unknown terrains. In such scenarios, the fixed infrastructure that could support communication is often destroyed or no such infrastructure exists. The agents themselves could form an ad-hoc network for communication. However, their number is often limited and the area to be explored is often very large. The presence of obstacles significantly reduces the line of sight communication distance. Hence the movements of robots get severely restricted, if they try to remain in a connected network.

Creating an AMRoNet layer that acts as an infrastructure to support the communication of the agents, would not restrict their movements. Thus we have a two tier network, with the agents and base stations lying at the upper layer and the AMRoNet *routers* deployed at the lower layer. AMRoNet could also provide various other services to the agents, such as location information, topological maps and shortest path to base stations, and assist the search and rescue operation of the agents. Whenever there is disconnection they could relocate and maintain the connectivity.

The swarm-based approach proposed in Chapter 4 tries to achieve optimal coverage in sensing range-based coverage problems by forming triangular grid structure with aimed inter-robot distance $D_{AIM} = \sqrt{3}r_s$ and creates a connected

network when $\frac{r_c}{r} \geq \sqrt{3}$ with $r = r_s$. However, in communication range-based area coverage problems where $r = r_c$, the strip-based structure shown in Figure 3.2 is optimal for achieving both full coverage and 1-connectivity. If we use the swarm-based approach with $D_{AIM} = r_c$ for area coverage, the triangular grid structure intended to form has a spatial density of only $D_{TRI} = \frac{1.155}{r^2}$ compared to the spatial density of the optimal strip-based structure $D_{STR} = \frac{0.536}{r^2}$.

Creating an optimal strip-based structure may not be a good solution if the environmental conditions are hostile, as the optimal structure has no room for automatic reconfiguration to regain coverage and connectivity in case of node failures or any changes in the environment. An over deployed structure that has redundant coverage area is more suitable in environments where failure of nodes are very common. In such environments the swarm-based approach would be an ideal choice as it allows faster relocation. However, in environments where such failures are uncommon and achieving the optimal configuration with least number of robots for full coverage and connectivity is very important; the swarm-based algorithm is no longer the best choice. Hence in this chapter, we focus on designing a new algorithm which also uses local rules, yet achieves a configuration that is close to the optimal solution. Here the main objectives are:

- Maximize the communication area coverage keeping the network connected.
- Use minimum number of robots in the deployment process.

5.1 Preliminaries

We have a two tier network when we use AMRoNet as an infrastructure layer to support the communication of the agents. The agents and base stations form the upper layer. The environment where the agents explore is a 2-D area denoted as A and has n base stations. There are N_a agents which are humans or robots capable of performing tasks such as urban search and rescue. As our focus is mainly on the AMRoNet, we do not specify the requirements of the agents and the base stations, which vary according to the scenario considered. The only assumption we make is that they have wireless devices to support communication.

The lower layer is the AMRoNet which consists of in total N_r routers. The routers denoted by R, are very simple robots compared to the agents with limited sensing capabilities with which they avoid obstacles and perform local navigation. Routers are equipped with wireless transceivers for communication.

We assume an isotropic radio communication range of r_c, where each node (agent, router or base station) can communicate with others located within a circle of radius r_c. We also assume that the communication area of one node, πr_c^2, is much less than A. Hence, the agents have to send packets over several routers to reach a particular destination (other agent or base station).

5.2 Related Work

Existing approaches to support the communication of the agents are mostly based on *mobile routers making a chain*. In [48,112] the authors present different strategies such as Manhattan-Hopper, Hopper, Chase explorer and Go-to-The-Middle, to maintain the connectivity of an explorer with a base station. In [170], depending on whether the knowledge of the agent's trajectory is available or not, the trajectories for the routers are estimated. These approaches maintain connectivity of the agents, if the routers move as fast as the agents. However, this assumption is not valid as the routers used to create AMRoNet are very simple robots and their speed is usually very small compared to the speed of the agents. The approaches in [48, 112] need routers that can move faster than the agents and the approach in [170] needs twice the speed of the agent, to keep the chain connected. Moreover, they cannot support connectivity of multiple exploring agents. Hence they are not useful in our scenario.

The self-deployment algorithms discussed in Section 3.2 or the swarm-based approach proposed in Chapter 4, though not meant for maintaining the connectivity of agents, could be used for creating the AMRoNet layer to facilitate communication between agents. Here also the spreading algorithms need router moving as fast as the agents to keep them connected. Using simple routers that are slower than the agents, the multi-robot spreading algorithms-based approaches work only if the deployment phase is finished prior to the exploration of the agents. However, in scenarios such as urban search and rescue, such proactive pre-deployment is not feasible.

5.3 Agent-assisted router deployment algorithm

We propose a greedy router deployment approach called *agent-assisted router deployment* for AMRoNet creation which does not need any fast moving routers or proactive pre-deployment phase. In agent-assisted router deployment, agents

carry routers during the exploration. They deploy routers greedily into the environment to those locations that maximize the local coverage maintaining connectivity. Routers move locally to maximize coverage. Such an approach is feasible, as our robots are very small and the agents can carry several robots during their exploration. A prototype system is shown in Figure 5.1, where the larger *Ranger* robots are used to deploy the mini-robot *Scout*, using a spring-based delivery mechanism [141].

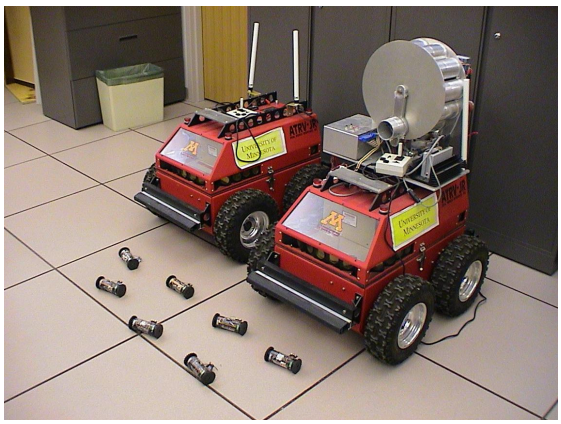


Figure 5.1: Prototype system for agent-assisted router deployment [141]

The agent-assisted router deployment algorithm given in Algorithm 5.1 has the following phases:

5.3.1 Initialization phase

In the initialization phase, the agents begin their exploration from the base stations. Each base station BS_i has a unique id i and has one node BN_i that acts as a base station server for all communications. The *group index* of BN_i is set to base station id i. Routers are denoted as R_{ij} and agents as A_{xy}, where the first subscript indicates their current group index and the second one indicates their unique id. The agents moving out of BS_i are initially connected to BN_i and other agents in the base station. Hence, each agent A_{xy} set its index x to i, the group index of its current reference node BN_i. All routers are initially *enabled* for a special deployment called triangular deployment discussed in Section 5.3.3.

5.3.2 Greedy deployment phase

The agents explore the area based on their own navigational algorithm. Figure 5.2 shows a schematic representation with two base stations and two agents (one agent

Algorithm 5.1 Agent-assisted router deployment algorithm

1. Initialization phase
for all $BS_i \leq n$ **do**
Set *group index* of BN_i to i
Set *index* of agent A_{xy} to the *group index* of current reference (BN_i)
Set all routers *enabled* for triangular placement
end for

2. Greedy and triangular deployment phase
/* Agents move autonomously to explore the environment */
if A_{xy} is about to lose connection with its only reference R_{ij} **then**
Deploy a new router with its index set to i
Set the position of the released router R_{ik} to A_{xy}'s current position
Update A_{xy}'s current reference to R_{ik}
else if A_{xy} enters the range of a router R_{pq} from the current reference R_{ij} **then**
if reference indices i and p are equal **then**
Update A_{xy}'s current reference to R_{pq}
else if R_{ij} or R_{pq} is enabled for triangular placement **then**
Release a new router to connect R_{ij} and R_{pq} with its index set to i
Instruct the released router R_{ik} to move to the goal point G
Disable R_{ij}, R_{ik} and R_{pq} for triangular placement
end if
end if

3. Local Coverage Maximization phase
if deployed router R_{ik} is instructed to move to the goal point G **then**
if goal location reached **then**
Stop navigation
else if obstacles encountered **then**
if connections with R_{ij} and R_{pq} are maintained **then**
Avoid obstacles
else
Stop navigation
end if
else
Goto goal location G
end if
else
Stay where deployed
end if

per base station) exploring an open area.

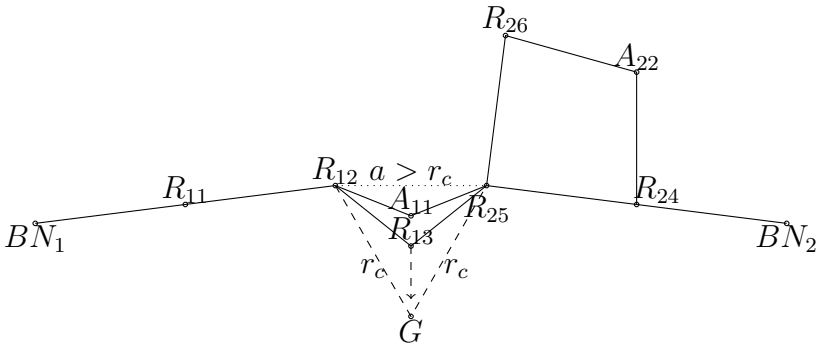


Figure 5.2: Schematic representation of agent-assisted router deployment in an open region

During the exploration, if A_{iy} is about to lose its connection to BN_i, it places a new router with its index set to i and position set to A_{iy}'s current position. The new router R_{ik}, for any $k \leq N_r$, is placed very close to the current location of A_{iy} in the direction towards current reference BN_i. This ensures that R_{ik} released is always connected to the current reference. R_{ik} becomes the new reference for A_{iy} and for all other agents within R_{ik}'s communication rage.

During the navigation, A_{iy} may move out of R_{ik}'s communication range and enter the range of a router R_{pq} for any $p \leq n$ and $q \leq N_r$ that has already been deployed. In this case R_{pq} becomes A_{iy}'s current reference. A_{iy} asks R_{pq} for its index and updates its index to p and becomes A_{py}. The agent repeats the placement steps when it is about to lose its connection to its current reference. If an agent has wireless links to many reference robots, any one of them acts as the agent's current reference. The agent releases a new router only when it loses connection to the last reference node in its communication range. We call this placement strategy as *greedy deployment*.

5.3.3 Triangular deployment phase

The greedy agent-assisted router deployment builds a graph G with the nodes at the base stations and with routers released during agents' exploration as its vertices. Agents exploring from one base station form a connected component, denoted as CC, of G. However, such CCs created from multiple base stations are not connected. When an agent A_{xy} enters into the range of R_{pq} from the current reference R_{ij}, for $i \neq p$ and $i = x$, CC_i and CC_p are temporarily connected. During the navigation, if A_{xy} loses it connection to R_{ij} but still has connection to

R_{pq}, A_{ij} does not place another router, as it has R_{pq} as its current reference. In this case, A_{xy} loses connection to its previous base station (BS_i), and therefore CC_i and CC_p get disconnected again.

To solve the disconnection problem, in such situations we adopt another deployment strategy called triangular deployment. In triangular deployment, when an agent A_{xy} encounters R_{pq} from the current reference R_{ij}, for $i \neq p$ and $i = x$, it releases a new router R_{ik}, for any $k \leq N_r$. This new router permanently connects CC_i and CC_p.

5.3.4 Local Coverage Maximization phase

In order to maximize the local coverage, the newly released router R_{ik} could move to a goal point G which still keeps R_{ij} and R_{pq} connected and also improves the local coverage. The goal point can be calculated as follows: If a is the distance between R_{ij} and R_{pq}, the goal point lies at a distance $\sqrt{r_c^2 - \frac{a}{2}^2}$ from the midpoint of the line joining R_{ij} and R_{pq} on the same side of the agent as shown in Figure 5.2. During the *goto goal* behavior, if the new router encounters an obstacle that cannot be avoided in few steps, it stops navigating to the goal location, as the obstacle could be too large to overcome without disconnecting R_{ij} and R_{pq}.

5.3.5 Optimization of triangular placement

To optimize the number of robots used during the triangular deployment, we propose two strategies. The first one needs global communication and the second one needs only local communication.

In the global strategy, when an agent A_{xy} entering into the range of R_{pq} from the current reference R_{ij} with $i \neq p$ for triangular deployment, it first checks with R_{ij} and R_{pq} if CC_i and CC_p are already connected. If not, it performs the triangular deployment and connects CC_i and CC_p. The router connecting CC_i and CC_p sends a message to all references connected to it either directly or by multi-hop networking, informing the new connected components. All these references update the information about the connected components in G.

In the local strategy, the router deployed sets the references R_{ij}, R_{pq} and itself as disabled for further triangular deployment. When a router R_{xy} or an agent A_{xy} entering into the range of R_{pq} from the current reference R_{ij} with $i \neq p$, it checks if both R_{ij} and R_{pq} have already been disabled from triangular deployment. This ensures that CC_i and CC_p always get connected and prevents redundant

deployment at the locations of triangular placements.

5.4 Experimental Evaluation

We evaluate the proposed agent-assisted router deployment using a simulation based empirical analysis. We use the Player server and Stage 2D simulator for our experiments. We consider a square area of size 32×32 m^2, which maps the floor plan of our institute as shown in Figure 5.3 for evaluation. The agents are modeled as Pioneer2dx robots, routers as Bebot robots and base stations' reference robots as Amigobot robots playerstagemanual. All robots are equipped with WiFi modules for communication. The base station robots are located at the corner of the simulation environment and are immobile. The scenario shown in Figure 5.3

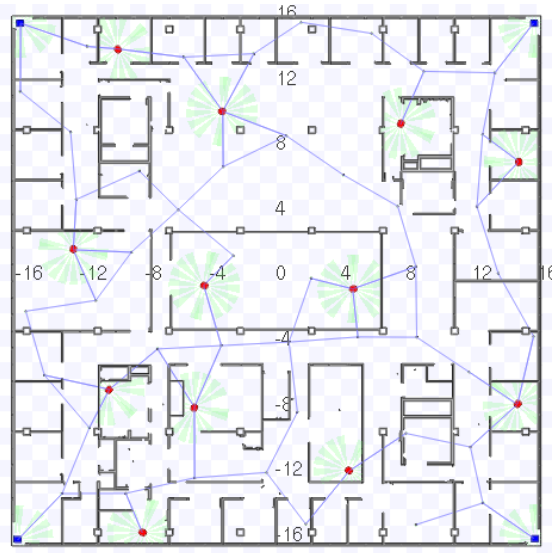


Figure 5.3: An example scenario with 12 agents and 4 base stations

has 4 base stations and 12 agents (3 per base station). The agents start their exploration from a point very close to the base station robots and are initially connected to them. We have chosen a random exploration strategy for the agents. They detect obstacles using their sonar sensors which have maximum range of 2 m and avoid them using the *obstacle avoidance* behavior implemented in the framework. The release of a new router by the agent is implemented by moving a router located outside the simulation environment to its placement point by the simulator. Routers released during the triangular placement use the *goto goal* behavior to navigate towards the goal points. They avoid collisions using their IR sensors which have maximum range of 20 cm.

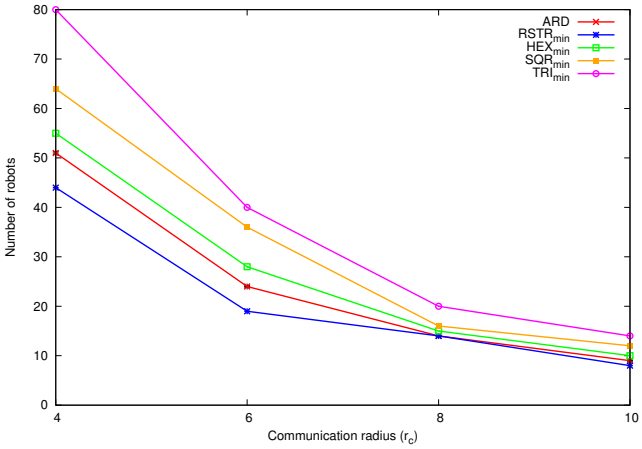


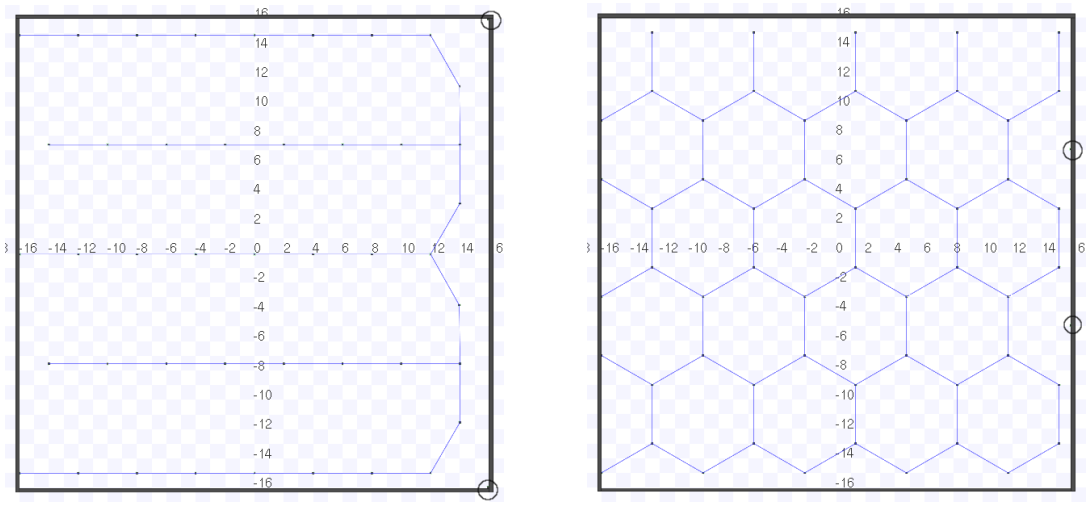
Figure 5.4: Comparison of deployed robot counts

5.4.1 Performance analysis

To analyze the performance of the agent-assisted router deployment algorithm, we vary parameters such as r_c and N_a. Figure 5.4 shows the result of the algorithm, when r_c is varied from 4 to 10 in a square area of size 32×32 m^2. The graph plot with label ARD shows the average number of routers (including the reference robot in the base station) deployed to cover the entire region, when all agents begin their exploration from one base station. Here, N_a is varied from 1 to 4. For each N_a, the simulation is repeated 5 times and the agents are assigned different start locations. So the graph plot with label ARD given in Figure 5.4 is the average of 20 simulations with confidence interval at 95% [42].

The expected number of robots required to cover an area by the static placement strategies of the commonly used regular patterns such as r-strip tile, hexagonal grid, square grid and triangular grid can be calculated using the spatial density of the patterns, i.e $d_{STR} = \frac{0.536}{r^2}$, $d_{HEX} = \frac{0.77}{r^2}$, $d_{SQR} = \frac{1}{r^2}$ and $d_{TRI} = \frac{1.155}{r^2}$. Since the area is bounded, the minimum number of robots actually required to cover the entire region is often higher than the expected values. This is clearly visible in the example figures, Figure 5.5a and Figure 5.5b, where the estimated (expected) number of robots needed for the r-strip $RSTR_{est}$ is 35 and the hexagonal grid HEX_{est} is 50, but the minimum required number for r-strip tile $RSTR_{min}$ is 44 and the hexagonal grid HEX_{min} is 55. The figures also show that there are still uncovered areas, e.g. the location of the robots highlighted with small circles. We cannot place additional routers to cover these areas, as they would be placed outside the specified area according to the regular placement pattern.

We compare the performance of our agent-assisted router deployment algo-



(a) R-strip tile based topology **(b)** Hexagonal-grid based topology

Figure 5.5: Static placement of regular pattern

rithm with the estimated number of robots required by an ideal agent-assisted static placement strategy. In the ideal agent-assisted static placement strategies, we assume that there is only one agent deploying the nodes and this agent moves in a controlled fashion so that it creates r-strip, hexagonal, square and triangular grid-based structures as desired. This assumption is made because any agent-assisted greedy deployment algorithms used for localized strip or grid structure creation would probably restrict the movement of agents or cause network disconnection.

The above issues are caused due to the following problems of such approaches: The routers released during the greedy deployment phase need to move to their goal points to create strip or grid, but they move very slowly compared to the agents. If the agents use these moving routers as their references, to prevent disconnections they may have to release new routers before the current references reach their goal points. Another problem is the presence of obstacles which prevents the routers from reaching the optimal goal point. A third problem occurs when we have multiple base stations. The pattern created from one base station may not be aligned with the other from another base station. There is also a problem that is specific to r-strip tile creation: instead of using only one router to connect two horizontal strips, if agents move in an adversarial manner, it may need one router per second router in the horizontal strip. Hence, the structure created by such algorithms may not be the expected strip or grid structure.

Figure 5.4 shows that the number of routers needed for agent assisted router

deployment algorithm (in red color) is quite close to the $RSTR_{min}$ value of the ideal agent-assisted static placement strategy. It is much lower than TRI_{min}, SQR_{min}, and HEX_{min} values. Hence it is one of the best localized approaches to create an AMRoNet, especially when we consider the fact that we compared the agent-assisted router deployment algorithm with an ideal algorithm performing static placement strategies.

If we calculate the expected number of robots needed for the hexagonal grid $HEX_{exp} = \frac{0.77*A}{r^2}$ in the specified square region for different r_c values and plot it, this curve would closely match the ARD curve shown in Figure 5.4. Hence we could use the equation $ARD = \frac{0.77*A}{r^2}$ to get an approximate estimate of the total number of routers needed to cover a given area A (if known in advance) by our agent assisted router deployment algorithm. This helps the agents in making an estimate on the number of routers they need to carry, before beginning their exploration.

5.4.2 Effect of number of agents and base stations

To analyze the effect of number of agents and base stations on the agent-assisted router deployment algorithm, we now vary the number of agents per base station N_{apbs} and the number of base stations n, for a fixed r_c. Figure 5.6 shows the average number of robots (including the base station robots) needed to cover the square area of size 32×32 m^2 for $N_{apbs} = 1, 2$ and 3, when n is varied from 1 to 4.

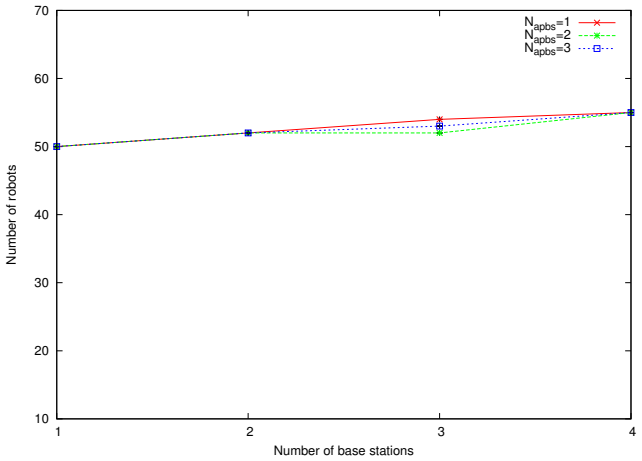


Figure 5.6: Effect of number of agents and base stations on the performance

Increasing the number of agents without increasing n do not affect the performance, as the deployments performed by the agents are based on the local rules

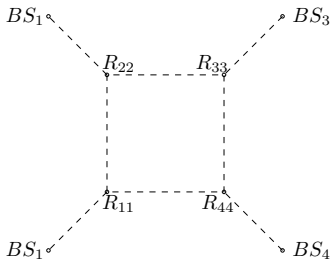


Figure 5.7: Redundant router deployment during local triangular deployment

which are in turn based only on losing or establishing connection with other routers and not with other agents. Hence the number of routers deployed is independent of the number of agents. The data points for a particular n shown in Figure 5.6 with different N_{apbs} confirm this.

Increasing the number of base stations may result in more triangular deployments. The total area covered by three robots in a triangular deployment is usually lesser than the total area covered by them in an optimal deployment. The largest overlap in a triangular deployment occurs when two references are separated by a distance slightly greater than r_c. However, such deployments do not increase the number of routers considerably. Even the greedy deployment may produce similar less optimal overlapping regions, e.g. when an agent connected to two references move out of the communication radius of both references simultaneously.

Figure 5.7 shows a scenario where three routers are released during the triangular deployment. Actually at most 2 routers are needed to make the four chains connected. Such redundant deployment increases with the number of base stations. We could add more local rules to make the increase bounded, but this is not actually needed as the agents move independently (in our experiments, they move randomly) and the structures similar to the one shown in Figure 5.7 occur very rarely. The graph plots for $N_{apbs} = 1$ and $N_{apbs} = 2$ depicted in Figure 5.6 also show that the total number of routers deployed is more or less the same for different base station counts.

5.5 Discussion

5.5.1 Merits of agent-assisted router deployment

From our experiments, we found that the proposed algorithms work well irrespective of the presence of obstacles in the area, where as in any localized approaches

for regular pattern creation their presence would cause severe problems. Our approach even works in areas where we do not have any prior model or map of the environment. It could be extended to make it work without any location information, in which case we require just the link quality estimate provided by the Wi-Fi devices. In such cases, the greedy deployment strategy is performed when the link quality drops below a threshold. Routers deployed during the triangular deployment, move in the direction where the link quality tends to be weak, in order to maximize the coverage area.

5.5.2 Self-spreading version

In those applications where a proactive pre-deployment of AMRoNet for supporting the agent communication is feasible, we could easily extend the agent-assisted router deployment algorithm into a self-spreading algorithm. There instead of agents deploying the routers, the routers themselves navigate and deploy. The routers could perform random-walk with an obstacle avoidance algorithm and on reaching the points that maximizes local communication area coverage, they could stop their navigation and become references for others to spread further.

We have tested a preliminary version of this approach and found that the final coverage achieved by this approach is better than the swarm-based approach presented in Chapter 4, as the swarm-based approach is not optimal for communication area coverage problems. However, the rate of coverage during the initial stages is much lower than the swarm-based approach. Combining swarm-based and self-spreading algorithms may achieve better performance.

5.6 Summary

In this chapter, we have presented a new localized and distributed algorithm for creating an ad-hoc mobile router network that facilitates communication between the agents without restricting their movements. The agent-assisted router deployment algorithm has a greedy deployment strategy for releasing new routers effectively into the area and a triangular deployment strategy for connecting different connected components created by the agents exploring from different base stations. Empirical analysis shows that the number of routers deployed by the agent-assisted router deployment algorithm is quite close to the number of routers needed for an ideal algorithm while placing nodes in the optimal r-strip tile pat-

tern in a bounded region. The performance of our algorithm is not affected by the number of agents or obstacles present in the environment. Increase in the number of base stations did not make any noticeable performance difference either.

CHAPTER 6

Self-optimizing Network

Self-optimization capability is one of the key features of self-organizing networks which enables the network to adjust regularly in varying loads and route efficiently in large-scale networks, especially when topology changes frequently. In our work we concentrate on the routing aspect of the self-optimizing networks. Routing refers to the most common *unicast* routing in our terminology, which involves routing of data packets from a single source to a single destination. Moreover, we are interested in routing protocols that provides guaranteed delivery, which means that if there is a path in the network between source and destination, the packets are always successfully delivered. We assume that the network is provided with an ideal medium access control (MAC) layer [167] where packets are not lost during forwarding steps.

Traditional routing protocols used for the wired networks cannot be used directly in ad-hoc wireless networks due to the highly dynamic network topology, absence of infrastructure for centralized administration and resource constrains. There have been a large number of routing protocols proposed for wireless ad hoc networks since 1980s and the design choices and approaches varies considerably. The MANET routing protocols focusing on distributed, low overhead and self-configuring routing aspects are of particular interest to AMRoNets. In this chapter, we briefly look at various routing protocols and focus on the protocols which are useful for AMRoNet routing.

Routing protocols could be classified into several types based on the structure, state information, cast property etc. [132]. From the perspective of information exploited, the existing routing protocols can mainly be classified into two categories: topology-based routing and geographic routing (also referred as location-based,

position-based and geometric routing in the literature) [132].

6.1 Topology-based routing

Topology-based routing protocols are based on the topological connectivity information, which basically uses the information about the links between the nodes to establish and maintain end-to-end paths for routing. Based on the routing information update mechanism the topology based routing could be classified into proactive (or table-driven), reactive (or on-demand), and hybrid approaches [132].

Proactive approaches maintain network topology information in the form of routing tables by periodically exchanging routing information and hence the routing information is readily available when it is needed. Proactive protocols such as Destination Sequenced Distance-Vector (DSDV) [142] and Optimal Link State Routing (OLSR) [95] has been proposed for ad-hoc networks by modifying the classical routing protocols such as Distance-Vector [39] or Link State [129] used in packet switching networks. Other prominent proactive protocols are Wireless Routing Protocol (WRP) [133], Cluster-head Gateway Switch Routing Protocol (CGSR) [37] and Source-Tree Adaptive Routing Protocol (STAR) [70].

The reactive algorithm does not maintain the topology information but gathers it only when it is needed such as at the beginning of a data session or in an existing route failure. As they typically require a route discovery process some delay is incurred before the packets are exchanged. The most prominent reactive routing protocols are Dynamic Source Routing (DSR) [97], Ad-hoc On-demand Distance Vector (AODV) routing [144] and Temporally Ordered Routing Algorithm (TORA) [139].

Hybrid algorithms combine the best features of these two categories. For routing within certain regions or zones a table driven approach is used and for beyond this zone an on-demand approach is used. Core Extraction Distributed Ad-hoc Routing (CEDAR) [160], Zone Routing Protocol (ZRP) [76] and limited-radius variants of DSDV [143] are some of the hybrid approaches.

Topology-based routing protocols could also be classified into flat and hierarchical protocols based on whether there is a logical hierarchy in the network or not. Hierarchal State Routing (HSR) [93] , Fisheye State Routing (FSR) [93], CGSR are good examples of hierarchal routing protocols whereas DSR, AODV and OLSR are flat routing protocols.

Topology based routing has significant overhead in establishing and updating

path information (routing table) between source and destination when topology changes frequently. It also has scalability issues as the overhead increases with increase in the number of nodes in the network which in turn increases the size of the routing table and the number of control packets sent to update path states [34].

6.2 Geographic routing protocol

Geographic routing protocol makes use of the geographic information of nodes (i.e., actual geographic coordinates or virtual relative coordinates) to deliver a message to its intended destination [34]. Unlike the traditional topology based routing which uses network addresses and routing tables to make routing decisions, here a forwarding node decides the next hop node by considering the geographic location of its neighbor nodes. Any changes in the network topology, like link or node failures, require routing information updates only at the nodes located close to that change. Traditional topology-based routing in contrast requires global routing information updates in such cases. This makes the geographic routing highly scalable and an attractive solution for large scale wireless ad hoc networks such as AMRoNets.

The main requirement for geographic routing is a positioning system. With the availability of small, inexpensive, and low-power Global Positioning System (GPS) receivers [85] or other localization systems [84], location information is available in wireless ad-hoc networks. As AMRoNet nodes are mobile robots, position information is available at-least from the dead-reckoning navigation systems [158]. An additional requirement for geographic routing is the knowledge of the destination location. In many applications especially sensing and monitoring tasks, destination is a base station node which is often fixed. Hence it is easy to obtain the destination location information. When the destination location is not known in advance, nodes use a *location service* [62], which provides a mapping of node addresses to their physical locations. Location service is often considered independent of routing in literature. Further information is available in [62, 161]. We now look at some of the most prominent geographic routing protocols in detail in the following sections.

6.2.1 Greedy routing

The basic idea of geographic routing was developed in the 1980s for packet radio networks [86, 166]. In these works, nodes forward messages according to locally

optimal forwarding rules and hence named as greedy forwarding or greedy routing. The greedy routing uses the positions of the sender, its neighboring nodes, and the destination to move the packet further towards the destination at every hop. The next hop nodes selected varies on the optimization objectives such as distance, direction and energy [34].

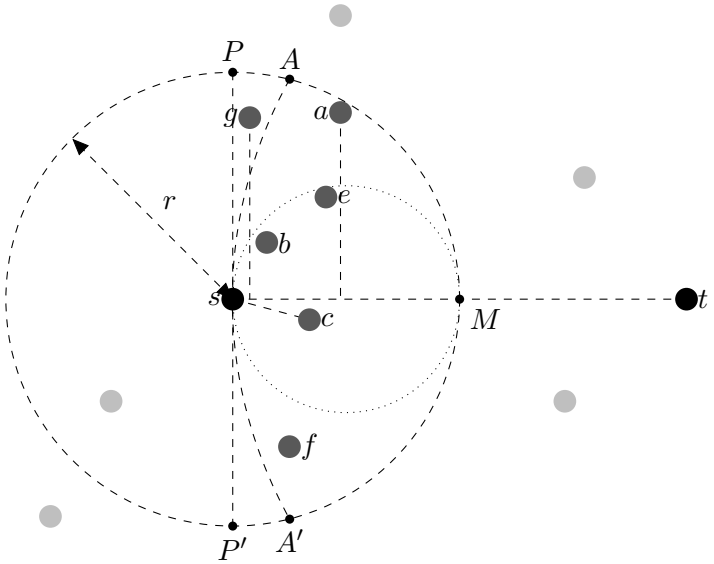


Figure 6.1: Greedy routing

Two most commonly used distance-based optimization criteria are progress and advance [34]. Progress is defined as the orthogonal projection of the line connecting the sender and the next-hop node onto the line connecting the sender and the destination. The region with positive progress for the source node s towards the destination (target) node t given in Figure 6.1 is $P - M - P' - P$. Advance denotes the distance gain towards the destination and is defined as the difference in distance between the sender and the destination and distance between a neighbor and the destination. The region with a positive advance for s towards t in Figure 6.1 is $A - M - A' - A$. In the example shown in Figure 6.1, node a is selected based on *Most Forward within Radius (MFR)* scheme (maximum progress) [166], node b is selected based on *Nearest with Forward Progress (NFR)* scheme [86], node c is selected based on *Most Advance within Radius (MAR)* scheme [54] and node b is again selected based on the *Nearest Closer (NC)* (nearest node with positive advance) scheme [163]. NFR and NC schemes are proposed based on the observation that energy consumption can be reduced by using short links.

Usually, the next hop nodes are chosen among the neighbors with a positive progress or with a positive advance. The most widely used forwarding strategy is the *greedy* scheme proposed in [54]. It selects the neighbor node which minimizes the Euclidean distance to destination which is equivalent to the MAR scheme which selects the neighbor node that has the greatest positive advance. Not all positive progress based schemes guarantee loop-free paths [34]. However selection based on maximum progress such as MFR or any positive advance, guarantees loop-free paths.

The direction based criterion select the next-hop node that tries to minimize the angular separation with respect to the destination. In the example given in Figure 6.1, node c is selected based on the criterion defined in *compass routing* [106], i.e. choose next hop node that has the direction closest to the line between the sender and the destination. However, this scheme is also not loop-free.

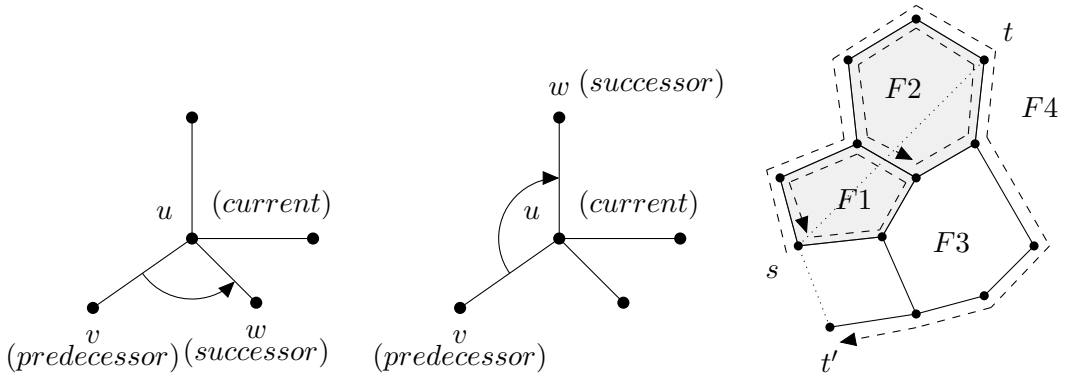
Energy-aware forwarding criterion select the next-hop node for which the total amount of power consumed for transmission from source to the next-hop node and from the next-hop node to destination is minimized. NC is such an energy aware scheme. However, same node might be selected by many routing tasks, which depletes the energy very fast leading to node failures. In order to maximize the number of successful routing tasks a power-cost metric considering the remaining battery power information along with the transmission power factor is introduced in [163]. Other criteria such as Maximum Weighted Progress (MWP) [53] which selects the neighbor with maximum advance per unit transmitted power and Energy-Advance-Random [31] which considers advance, residual energy and a random value for selecting the next hop node are considered in the literature.

The greedy forwarding schemes described so far require the locations of 1-hop neighbors which is usually acquired by a regular exchange of beacon ("hello") messages containing own position information [152]. However, in contention-based or beacon-less routing [64, 81], such beacon exchanges are not required. In such cases, the forwarding node cannot select the next-hop node explicitly, but the next-hop node can select itself in a contention with other neighbors. The basic principle is the same: the source broadcasts the message to its unknown neighbors and the neighbors located in a *forwarding area*, an area closer to the destination and where all nodes can overhear each other (e.g. dotted circle in Figure 6.1), contend for the message. They set a timer in accordance to their distance to the destination. So the node closest to the destination (most advance) has the shortest timeout and it retransmits the packet first. The remaining nodes overhear

the re-transmission and cancel the scheduled packet.

6.2.2 FACE routing

FACE routing also referred to as planar graph routing, originally proposed in 1999 under the names "Compass routing II" [106] and "FACE-2" [26], was the first geometric routing algorithm that guarantees message delivery if applied on a planar embedding of the communication network. A *planar embedding* is a graphical representation in a plane where edges intersect only at their end points. The edges of a planar graph constitute polygons which partition the plane into one outer and several inner faces as depicted in Figure 6.2c.



(a) Left hand rule **(b)** Right hand rule **(c)** Faces

Figure 6.2: FACE routing rules

The basic idea of planar graph routing is to forward messages along the interiors of a sequence of adjacent faces providing progress towards the destination as depicted in Figure 6.2c. FACE routing is based on face traversal and face changing rules [152].

Face traversal of a single face can be done in a localized way by applying the well-known left or right hand rule used for exploring mazes. The rule sates that, if the maze is simply connected, by keeping one hand in contact with one wall of the maze the player is guaranteed not to get lost and will reach a different exit if there is one; otherwise the player will return to the entrance [152]. Considering the link order during face traversal, right-hand or left-hand rule is similar to sending the message from the current node u along the edge uw which is lying next in clockwise or counterclockwise direction from the previous visited edge vu as shown in Figure 6.2a and 6.2b respectively. However considering the path taken during

face traversal shown in Figure 6.2c, right-hand rule follows the boundary of an inner face in the counterclockwise direction and an outer face in the clockwise direction as depicted by the dotted line in the figure.

Face changing rules determine the condition for switching from one face to the adjacent face. Face changing rule has different variants in literature. In the Compass Routing II (FACE-1 routing), the rule is stated as follows: During the face traversal the algorithm finds the *exit point* where the edge intersects the line connecting the source and destination and which is closest to the destination compared to all other intersections encountered. After traversing the whole face, it goes back to that particular exit point and switches to the next face. Figure 6.3

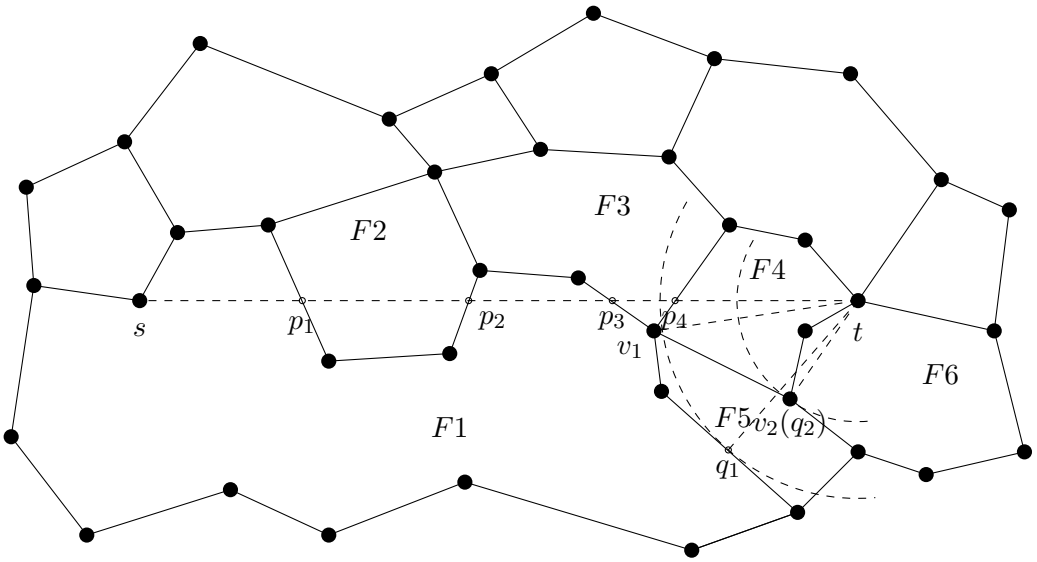


Figure 6.3: FACE routing illustration

shows an example to illustrate FACE-1 routing protocol. Here a packet is sent from source s to destination t. The first face for traversing is F_1, the face that contains s and intersects $s-t$ line. On traversing along the boundary of face F_1 according to the left hand rule, it finds the intersection points p_1, p_2 and p_3 and completes the traversal. As p_3 is the closest point to t, the packet traverse F_1 again to reach the point p_3. It is better to consider the shortest among the two possible paths to reach the closest point. Now the FACE routing is repeated by considering p_3 as the new s. So a face change to F_3 occurs, as it is the new face intersecting $s-t$ (p_3-t) line and containing p_3. On traversing this face according to the left hand rule it finds p_4 as the closest intersection, switches to face F_4 and reaches the destination node t located in this face.

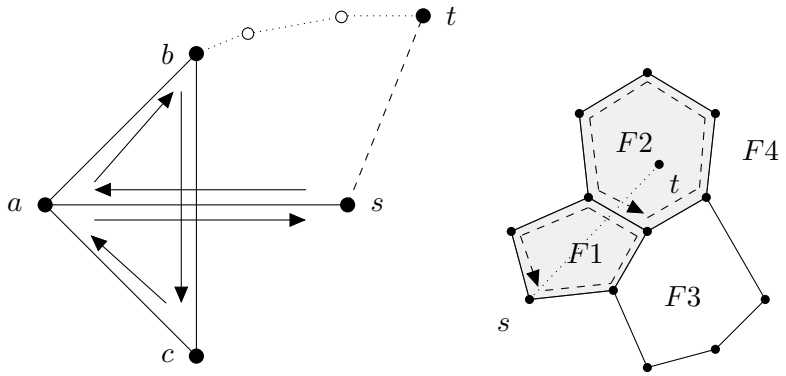
FACE-1 routing terminates on any simple planar graph in O(n) steps, where n is the number of nodes in the network. To speed up the FACE-1 routing, FACE-2 routing has been proposed. In this algorithm it does not search for the whole face for the best exit point, but switches the face on encountering the first exit point and thus avoids traversing huge faces that point away from the destination. Though this algorithm is practically more efficient than FACE-1 routing, the theoretical worst case is worse than the FACE-1, i.e. $O(n^2)$. In the example given in Figure 6.3 FACE-2 algorithm traverse along the boundary of face F_1 according to the left and finds p_1 as the first intersecting point. It changes to face F_2 at this first exit point, as it intersects $s-t$ (p_1-t) line and contains p_1. On exploring the face F_2, it finds the next new intersecting point p_2. It changes back to the face F_1, explore F_1 and finds the next new intersecting point p_3. It moves to face F_3, explores F_3 and finds the next point p_4. It advances to face $F4$ and finally reaches the destination t.

The above face routing protocols are classified as *continuative* strategies [60], where they keep the line $s-t$ as a reference for the whole face routing process. However, in *volatile* strategies [60] the reference line changes at each node where the face change occurs. In these FACE routing variants, the line connecting these nodes and the destination is considered as the new reference. The face traversal is restarted at these nodes. An example for volatile FACE routing is Other FACE routing (OFR^+) [108], where instead of finding the intersecting point on $s-t$ line located closest to destination, it finds the node located closest to the destination. In the example given in Figure 6.3, after a complete exploration of face F_1, OFR^+ finds the closet node v_1. The packet traverse again F_1 to reach v_1 and the face routing is restarted at this node. The reference line changes to v_1-t. Now the face routing changes to the next face F_4. On traversing F_4, it reaches the destination t. The authors also propose a variant of this rule OFR, where instead of sending the message to the closest node, it is sent to the closest point on the boundary of the current face towards destination [107]. In the example given in Figure 6.3, the closest point on the boundary of face F_1 is q_1. OFR switches to the face F_5 containing q_1 and intersecting the q_1-t line. The routing is then restarted. In F_5, the closest point q_2 is the node v_2. Routing switches to the new face F_6 and finally reaches the destination t.

FACE routing protocols varies according to the face changing rule as explained above. Face changing rule is also a critical condition which determines whether the algorithm provide delivery guarantee in arbitrary planar graphs or only in a certain types of planar graphs. Compass Routing II (FACE-1), FACE-2 and

OFR routing provide delivery guarantee on arbitrary planar graphs, whereas the FACE routing proposed in Greedy Perimeter Scale Routing (GPSR) [100], OFR$^+$, Greedy Path Vector Face Routing (GPVFR) [117], does not guarantee delivery on arbitrary planar graphs [60].

The planarity and connectivity of the network graph are very important for assuring delivery guaranty. Crossing links can cause detours and routing loops as shown in Figure 6.4a where the intersecting links causes FACE routing failure. A packet send from s destined to t follows the path $s-a-b-c-a-s$ according to the left hand rule and reaches back to the node s. To avoid routing loops, FACE routing remembers the first edge of the current face exploration and when the first edge is traversed once again, the message gets dropped. Hence the packet gets dropped in this example. There are also scenarios where there is no path from



(a) Cross links **(b)** Disconnection

Figure 6.4: FACE routing failures

source to destination due to disconnections in the network. Figure 6.4b shows an example where the destination t is disconnected from the rest of the network. On planar graphs, the message will reach the face containing the destination node (face F_2 in the example) and traverse the face without detecting another face providing progress towards the destination node. In such cases on reaching again the first edge of the face traversal, the message gets dropped.

6.2.2.1 Improved FACE routing

FACE routing can be performed locally like the greedy routing with additional information such as the first edge traversed on the current face and the position of the last intersection where a face change occurred. Choosing the appropriate traversal direction while starting a face traversal is important for efficient oper-

ation. However, finding the appropriate direction is impossible just with local information. In improved FACE routing the number of edges of each face that must be traversed can be bounded by restricting the search area.

Bounded Face Routing (BFR) [109] is an improved face routing that restricts the search area of face traversal by a bounding ellipse E. Let the upper-bound on the Euclidean length $c_d(p^*)$ of a shortest path p^* from s to t be $\hat{c_d}$. The ellipse E with foci s and t and the length of the major axis set to $\hat{c_d}$ will contain the shortest path from s to t. As in the FACE-1 routing, BFR begins the exploration of the face in one of the two possible directions (clockwise or counterclockwise) and remembers the best exit point p during the traversal. Whenever the packet hits the ellipse during the face traversal, it turns back and continues its exploration of the current face in the opposite direction until the ellipse is hit for the second time. The face traversal is finished when the ellipse is hit for the second time, or the traversal reaches the starting point without hitting the ellipse as in the normal FACE routing. After the traversal, the packet is sent to the closest exist point p found during the traversal as in the FACE routing and the routing switches to the new face. As the number of times an edge is traversed is bounded by a constant, and the number of edges in the ellipse is bounded by $O(|st|^2)$, the cost of BFR is bounded by $O(\hat{c_d}^2)$.

If the exploration of the current face does not yield a better exit point p, BFR does not find a route to the destination. This is because the upper-bound on the length of the best route is unknown. To avoid such situation an Adaptive Face Routing (AFR) [109] has been proposed, where the estimate $\tilde{c_d}$ for the unknown length of the shortest path $c_d(p^*)$ is initialized to twice the Euclidean distance between s and t, i.e. $2|st|$ and BFR is executed. If BFR does not succeed, the estimate for the length of the shortest path is dynamically increased (doubled), i.e. $\tilde{c_d} = 2\tilde{c_d}$. If s and t are connected, AFR will finally find a path to t. The cost of AFR is dominated by the routing steps taken in the final ellipse and consequently it is also $O(c_d(p^*)^2)$. In the same way how AFR improves FACE Routing, Other Adaptive FACE Routing (OAFR) improving OFR has also been proposed [107].

6.2.3 Void handling

Greedy routing algorithm if successfully delivers a packet, the routing paths they used has a weight that is comparable to the weight of the shortest possible path. In greedy routing variants, even if forwarding ends up at a node which is the best

compared to its neighbor nodes, further forwarding from these nodes might be impossible according to the forwarding metric, even though a path from source to destination exists. This is depicted in Figure 6.5 where none of the neighbors

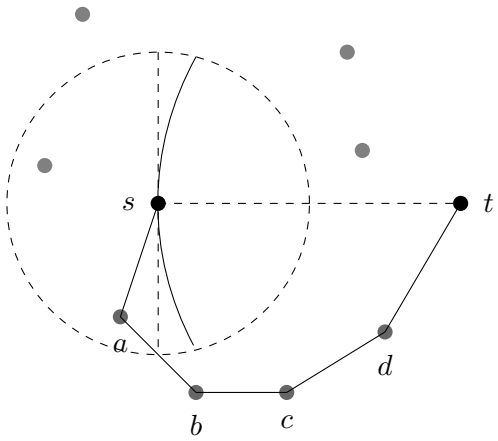


Figure 6.5: A communication void where greedy routing fails

of the sender s can be selected as the next-hop node according to the positive advance-based (progress-based) forwarding metrics described in Section 6.2.1, but there exists topologically valid paths from s to t, e.g. $s - a - b - c - d - t$. This is commonly referred to as communication void or local minimum [34]. The simple greedy routing drops the packet in such situations and the delivery fails. To solve this problem, an alternate mode is needed for the greedy routing which is commonly called as void handling or recovery mechanism [34].

A simple recovery strategy would be to allow the packet to travel in the backward direction for one hop from the void node and then forward it again excluding the previously visited void node. For example in Figure 6.5, the source s sends the packet back to a and on excluding the previously visited node s, it finds the path $a - b - c - d - t$ to the destination t. This scheme is proposed in the *Geographical Distance Routing (GEDIR)* [162] algorithm and it is loop-free. There are also strategies proposed to improve the delivery rate of greedy routing such as selecting a suitable neighbor out of the 2-hop neighborhood and forwarding the packet to the direct neighbor which is connected to the selected node [162]. In the above example, b is the two hop neighbor selected for the source s and a will be the direct neighbor considered. However, such simple approaches cannot always overcome void nodes. We need void handling solutions which provide guaranteed message delivery.

6.2.3.1 FACE routing-based void handling

FACE routing when applied on a planar embedding of the communication network, guarantees message delivery. However, planarization process typically removes edges and often results in paths longer than the shortest path between source and destination. Using a combination of FACE routing and greedy routing is mutually beneficial for them. The most prominent combination is using FACE routing as void handling solution when the greedy routing fails at communication voids. It is the only known localized single path approach to recover from such greedy routing failures [60].

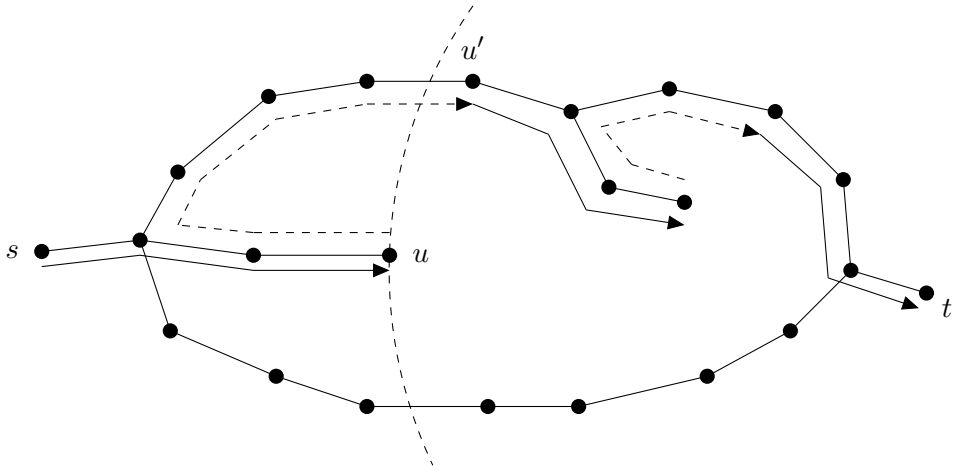
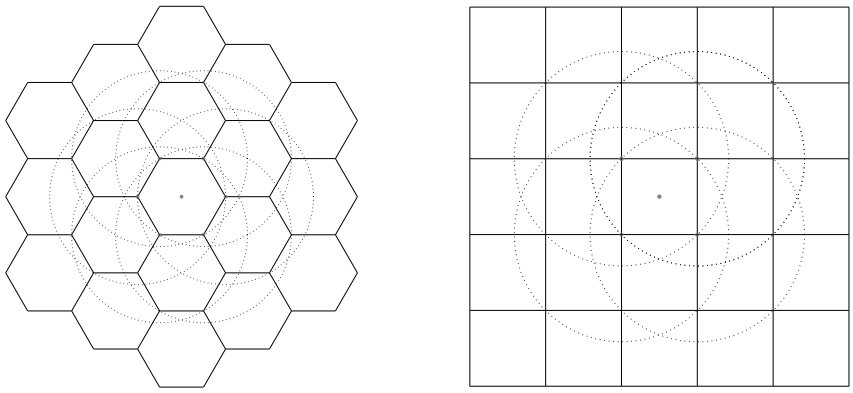


Figure 6.6: Greedy-FACE-Greedy routing

FACE routing may return to greedy routing when the current node's or its neighbors' distance to the destination node is smaller than the distance between the node which initiated planar graph routing and the destination node. This results in a Greedy-FACE-Greedy (GFG) routing path. In example shown in Figure 6.6, for a source s and destination t a packet is routed in the greedy mode from s till node u where it encounters the communication void and then switches to the FACE routing mode. The packet is forwarded till u' which is closer to t than u and hence the routing switches back to greedy mode from u'. The solid line indicates the greedy routing path and dotted line indicated the FACE routing path.

In the greedy and FACE routing combination, when the face routing variant AFR is combined with greedy routing we get GAFR [107]. However, the asymptotic optimality of AFR is lost in this combination. To solve this problem, the authors propose OAFR or the variant OAFR*, which when combined with greedy



(a) Hexagonal grid based overlay graph

(b) Square grid based overlay graph

Figure 6.7: Overlay graphs

routing gives GOAFR or GOAFR* [107] which are both average-case efficient and asymptotically optimal. We have seen in Section 6.2.2 that Compass Routing II (FACE-1), FACE-2 and OFR* provide delivery guarantee on arbitrary planar graphs. Their combination with greedy routing also provides this delivery guarantee [60]. Most of the greedy and FACE routing combinations return to greedy routing as soon as possible, but considering other rules before switching to greedy routing shows better performance [108].

So far we have seen FACE routing applied directly at the planar network graphs for recovering from greedy routing failures. There are other approaches where FACE routing is applied at *virtual overlay*-based planar graphs [57, 58, 123, 146, 169]. In these approaches, a proactive construction of planar overlay graphs is carried out. The overlay graphs are created as follows: The plane is partitioned into an infinite mesh of regular polygons such as square or hexagonal grids called geographic clusters, with the grid diameter set to the transmission range of the nodes as shown in Figure 6.7a and Figure 6.7b. Nodes are aggregated to such clusters based on their current location. Virtual nodes are created at the center of the clusters containing at least one network node. These virtual nodes constitute the overlay graph vertices. Two clusters are considered as *adjacent*, if there exists at least one edge in the original network graph, with the end nodes located in those clusters. Virtual edges connecting virtual nodes are drawn between such adjacent clusters [58, 146].

Creating an overlay graph first and removing intersecting links in the overlay graph to make it planar may cause disconnections and cannot guarantee message delivery [58, 146]. An alternate idea is to prevent intersecting links in the original

network graph by planarizing it first, create an overlay graph from this planarized subgraph. However, such overlay graphs are not planar and to planarize these graphs, intersecting links are removed locally. To preserve connectivity during such link removals virtual edges are added. This approach guarantees planarity and connectivity of overlay graphs [57].

6.2.3.2 Other void handling mechanisms

There are several other void handling mechanisms that do not use FACE routing variants for recovery. A comprehensive overview is provided in [32,34]. We provide here a brief overview of some of the prominent approaches.

One approach that exploits the geometric properties of voids is presented in [50]. Each node exploits the geometric properties of neighbors to locally detect the possibility of being a void node. It then initiates the BoundHole algorithm that uses greedy sweeping to set up a path around the void regions such that when a packet in greedy mode arrives at the void node, it could use this precomputed path. BoundHole algorithm does not guarantee packet delivery [34].

Another approach is link-reversal-based void handling [34]. In the presence of communication voids, the network graph becomes a destination-disoriented DAG as the void nodes have no outgoing links. Link-reversal-based approaches try to transform a destination-disoriented Directed Acyclic Graph (DAG) into a destination-oriented DAG by reversing the directions of the links. When the void node reverses the directions of all incoming links it is a full reversal method, otherwise it is a partial reversal method. Cost-based forwarding in Partial-partition Avoiding Geographic Routing-Mobile (PAGER-M) [186] is an example for full reversal algorithm and Distance Upgrading Algorithm (DUA) [36] is an example for partial reversal algorithm. This is a localized void handling solution with guaranteed delivery. However, it is meant for networks with nodes delivering their data packets to a fixed destination (base station).

A proactive construction that creates hull trees for void handling is presented in [116]. The associated routing protocol Greedy Distributed Spanning Tree Routing (GDSTR), switches to routing on a spanning tree instead of routing on planar faces. Each node stores the convex hull of all node locations in the subtree and also the convex hull information of its descendants. When greedy routing fails the algorithm searches the descendants of the stuck node and if unsuccessful it passes the packet to the parent towards the root. The packet is routed upwards in the tree until a node is found, whose convex hull contains the destination and

then routed downwards in the tree until the destination is reached. However, this approach is not localized.

There are also topology-based void handling solutions which rely on topological connectivity information to overcome voids. Protocols using topology-based void handling solutions are hybrid routing protocols in a strict sense instead of pure geographic routing protocols. It uses partial flooding [33] or depth-first search [164] techniques to discover a path to another node where greedy forwarding can be reactivated. Cartesian Routing [54], one-hop flooding [162], Partial Source Routing (PSR) in On-demand Geographic Forwarding (OGF) [33] and Partial Hop-by-hop Routing (PHR) in Geographic Routing Algorithm (GRA) [164] are some of the prominent topology-based void handling solutions.

There are many heuristic void handling solutions such as passive participation [185], active exploration [30, 53] and also hybrid solutions that combine two different void handling solutions such as BoundHole plus restricted flooding [51] and active exploration plus passive participation [30].

6.3 Hybrid routing

Hybrid Routing combines topological and geographic routing. Algorithms such as LAR [104] makes use of location information for improving topology based routing. LAR designates two geographical regions, an *ExpectedZone* in which destination node is expected to be present and *RequestZone* where the path-finding control packets are propagated, for selective forwarding of control packets. The flooding in LAR is restricted to small geographical region.

Topology-based routing that uses location information for creating grids and routing on the overlay grids has been reported in [96, 121]. In the Zone-based Hierarchical Link State Routing Protocol (ZHLS) [96], physical area is partitioned into a number of squares grids called *zones* and nodes are associated to their respective zones based on their geographic location. Routing is then performed in a two-level manner, i.e. an intra-zone routing and an inter-zone routing. Each node in a zone is aware of other nodes in the zone and the neighboring zones using links state packet exchange (a proactive protocol). Using *zone link state* packet, links changes between zones are updated. Thus, any intra-zone link state change will be propagated to all other nodes in the zone, and any inter-zone link state change to the whole network. In the GRID protocol [121], one mobile host (if any) will be elected as the leader of the grid and routing is performed in a grid-by-grid

manner through grid leaders. As long as there exists a leader in each grid that constitutes the route, the route is considered alive. If a leader leaves its original grid, a "handoff" procedure occurs and the leader will pass the routing table to the next leader. Only grid leaders are responsible for route searching and avoids broadcast storm problem.

Segment-by-Segment Routing [29] is also a similar approach considering the grid formation. The cluster heads maintains a k-hop vicinity routing table in the overlay graph for remote routing and an inter cluster routing table for local routing. The k-hop information is used in the greedy forwarding to select the most progress node and at voids the right hand rule is used to circumvent holes boundary.

Landmark-based or anchor based geographic routing is presented in [24, 52] where a global map of the anchors are discovered and maintained by the source or obtained from the global information such as maps for greedy-packet-forwarding along anchored paths. In Terminodes [24], packets are routed according to Terminode Local Routing, a proactive distance vector scheme, if the destination is close to the sending node. For long distance routing, it uses Terminode Remote Routing, a greedy-packet-forwarding approach along an anchored path. The set of anchors form a route vector for greedy forwarding without falling into voids. The anchored paths need to be discovered and maintained by the source, as in FADP protocol or obtained from the global information such as maps, as in GMPD protocol [24].

In GLIDER [52], a set of landmarks is selected. For each landmark, a *tile* is created which is a set of points that are closer to this landmark than to any other landmark. An overlay graph with tiles as graph nodes and tile adjacency as graph edges is created. Routing consists of two steps: 1) at each hop the overlay graph is consulted to determine the next tile and 2) the neighbor that is closest to the landmark of the next tile is chosen as next hop. On encountering void nodes, the intra tile routing switches to tile flooding.

6.4 Summary

We have seen some of the most prominent routing protocols proposed for ad hoc networks. Among these protocols, considering the distributed, low overhead and self-configuring routing aspects, geographic routing is an interesting solution. The localized and stateless nature of the protocol makes it highly scalable and useful for

large scale wireless ad hoc networks such as AMRoNets. The main requirement for geographic routing is a positioning system which is readily available in AMRoNet nodes as they are mobile robots. The basic geographic routing has a greedy routing mode which forward messages according to locally optimal forwarding rules, but packets might get stuck at void regions. Using planar graph-based routing (also known as FACE routing) as a void handling solution keeps the stateless property of greedy routing and helps in overcoming void regions.

Greedy forwarding if successfully delivers a packet finds routing paths comparable to the shortest possible path, but it alone does not guarantee the delivery of packets because of void regions. FACE routing on a plane graph does guarantee the delivery of packets but often has paths longer than the shortest path. For improved performance, FACE routing is often combined with greedy forwarding and is used as the void handling solution to overcome dead-ends when greedy forwarding fails. It is the only known localized single path approach to recover from such greedy routing failures. Hence it is the ideal combination for a stateless and localized geographic routing.

CHAPTER 7

Graph Planarization

In Chapter 6 we have seen that geographic routing is a highly scalable and an attractive solution for AMRoNets. The basic geographic routing has a greedy routing mode which forward messages according to locally optimal forwarding rules, but packets might get stuck at void regions. Using planar graph-based routing (also known as FACE routing) as a void handling solution keeps the stateless property of greedy routing and helps in overcoming void regions.

In general, a wireless network topology is not a planar embedding. Hence, a planarization process is often required which typically removes a set of links and makes the graph planar. In large scale wireless networks, distributed planarization algorithms are indispensable and essential. Algorithms with low overhead in terms of additional computing or message exchange are highly desirable. Localized planarization algorithms where each node requires to know only its one hop neighbors or at least neighbors up to a limited number of hops, and where messages propagated for planarization are limited to the single hop or limited hop neighbors, are good choices for algorithms with low overheads.

7.1 Network Model

We model the wireless network as a graph $G = (V, E)$ embedded in two dimensional Euclidean space $\mathbb{R}^2$. The graph has a finite set of vertices V that corresponds to the nodes in the network. Each node knows its 2D position. The set of edges E corresponds to the wireless links between the nodes. An edge $e \in E$ consisting of vertices u and v, is denoted by $e = uv$, where u and v are the endpoints of the edge and are neighbors in the network. A path $P = v_1 v_2 \dots v_k$ is a non-null

sequence of vertices with $\forall i \in 1 \ldots k-1 : v_i v_{i+1} \in E$. A path with no repeated vertices is called a *simple path*. In our work *path* always refers to *simple path*.

The planar graph $G_p = (V, E')$ of a graph $G = (V, E)$ has the same vertex set V, but the edge set $E' \subseteq E$ due to the removal of some of the edges, during the planarization process. There are a few metrics to judge the quality of planarization algorithms.

- **Connectivity:** The edge removal during planarization should not cause network partitions, i.e. if there is a (multi-hop) path between any two nodes u and v in G, then there should also be some path between these nodes in G_p. Thus, *connectivity* is a mandatory property of the planarization process.

- **Planarity:** The objective of the planarization algorithm is to create a planar graph G_p from a given graph G. Most of the planarization algorithms work only if the input graph G obeys certain structural properties. Hence if we apply those algorithms in graphs which do not obey these properties, G_p may still contains intersecting links. Algorithms which have less restriction on the structural properties are highly desirable.

- **Spanning ratio:** Another metric to determine the quality of the planarization process is the *spanning ratio*, which is defined as,

$$S = \max_{u,v \in V} \frac{PL_{G_p}(u, v)}{d(u, v)},$$

  where $PL_{G_p}(u, v)$ is the length of the shortest path between u and v in the planarized graph G_p and $d(u, v)$ is the Euclidean distance between u and v. A graph is a c-spanner, if for all $u, v \in V$ there exists a path from u to v with $|p(u, v)| \leq c \cdot d(u, v)$ and it is a weak c-spanner, if this path is covered by a disk of radius $c \cdot d(u, v)$ centered at the midpoint of u and v [153]. The edge removal during planarization process may increase the path length between two nodes in planarized graph. Showing a spanner graph which is still spanner after the planarization process reflects the quality of the planarization process.

The most commonly used network model to model the topology of ad-hoc wireless networks is Disk Graph Model [98]. In this model an edge between two nodes u and v exists, iff the $d(u, v) \leq R$, where R is the radius of the disk. The special case where $R = 1$ is called the Unit Disk Graph (UDG) [40]. Here we

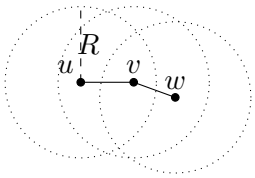


Figure 7.1: Unit Disk Graph

assume that transmission range of all nodes is set to a precise radius R and beyond this range communication is impossible. Figure 7.1 shows an example with three nodes u v and w with radius R. The edges uv, vw exists as $d(u, v)$ and $d(v, w)$ are less than or equal to R.

7.2 Graph planarization algorithms

In computational geometry, to create planar graphs for a given set of points P in $\mathbb{R}^2$, graph algorithms such as Relative Neighborhood Graph (RNG) [173], Gabriel Graph (GG) [65], and Delaunay Triangulation (DT) [44] are commonly used. Though there are many efficient and fast ways to find such graphs, most of them usually uses centralized approaches. In wireless networks, the focus is on distributed localized planarization techniques as they have less overhead in maintaining the planar topology, especially when the topology changes frequently.

The Relative Neighborhood Graph of a set of points P, RNG(P), contains all edges uv, $\forall u, v \in P$, such that the intersection of the two circles with centers u and v, and radii $|uv|$ contains no other node as depicted in Figure 7.2. Formally, $\forall u, v \in V : uv$ exists, $iff \nexists w \in V : max\{d(u, w), d(v, w)\} < d(u, v)$.

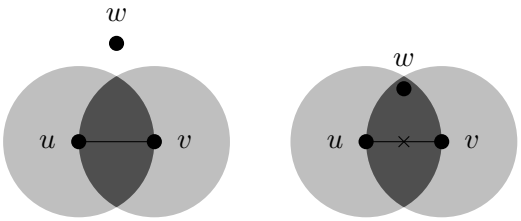


Figure 7.2: Relative Neighborhood Graph

RNG is not a c-spanner. The worst-case spanning ratio is $\Omega(|V|)$ [26]. Hence, nodes that are only a few hops apart in the original graph may become very distant in the RNG planarized graph. RNG(G) of a graph G can be easily constructed with local algorithms. Each node u keeps only those outgoing edges uv which satisfy the RNG condition. For input graphs which obey UDG property, RNG

planarized graph, RNG(G), is always connected if the original graph G is also connected.

The Gabriel Graph of a set of points P, GG(P) contains all edges uv, if the circle with diameter $|uv|$ and nodes u and v on its circumference contains no other node as depicted in Figure 7.3. Formally, $\forall u, v \in P : uv$ exists, $iff \nexists w \in V : d^2(u,w) + d^2(v,w) < d^2(u,v)$. GG can also be easily constructed with local

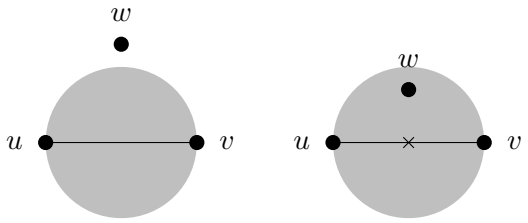


Figure 7.3: Gabriel Graph

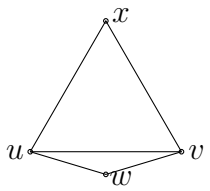
algorithms where each node u keeps only those outgoing edges uv which satisfy the GG condition. It also guarantees connectivity if the original UDG G is connected. The worst-case spanning ratio of GG is $\Omega(|\sqrt{V}|)$ [26].

The Delaunay Triangulation (DT) is the dual of the Voronoi diagram, and is widely used in various application areas. The main advantage of DT is that it is a spanner. The DT for a set of points P, DT(P), contains all triangles (u, v, w), $\forall u, v, w \in P$, such that the circumcircle of u, v, and w contains no other node. This is called the *empty circle* rule [44]. Constructing DT via empty circle rule requires global knowledge and localized construction is not possible. However, it is possible to locally construct Delaunay Triangulation-like topologies which are constant unit disk graph spanners, e.g. Restricted Delaunay Triangulation (RDT) [69] and Localized Delaunay triangulation (LDEL) [120]. LDEL is a spanner with factor 2.5. The basic approach used in these Localized Delaunay triangulation (LDT) is that each node computes the Delaunay triangulation of its 1-hop neighbors using the empty circle rule. From the subset of outgoing Delaunay edges, each node preserves all outgoing edges which are preserved by the node on the other edge end point as well.

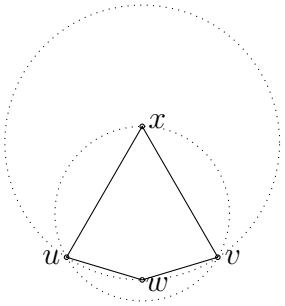
There also other localized planarization approaches in the literature. A localized construction based on spanning trees called Local Minimum Spanning Tree (LMST) is given in [119]. MST $\subseteq$ RNG $\subseteq$ GG $\subseteq$ DT [119]. They are also not spanners. A reactive planar spanner construction using angle-based and Delaunay-based *Direct Planarization* (DP) techniques is presented in [59].

7.3 Problems of implicit planarization

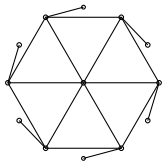
All existing localized planarization techniques [26, 59, 69, 100, 119] are implicit planarization, which means that links are removed irrespective of the fact whether an intersection exists or not. Figure 7.4b shows an example where the Localized DT-based algorithm (LDT) removes a link due to implicit planarization. These implicit planarization techniques provably produce planar graphs in re-



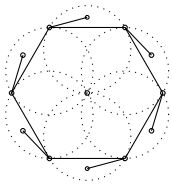
(a) A UDG modeled network



(b) LDT planarized graph removes a non-intersecting link



(c) A non-UDG modeled network



(d) LDT planarized graph causes disconnection

Figure 7.4: Implicit planarization algorithm

stricted wireless models such as UDG [40]. However in graphs that are not UDG, these approaches do not work, as depicted in the Figure 7.4d where LDT causes disconnection.

7.4 A new explicit planarization

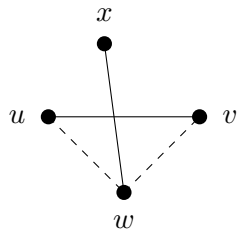
We solve the non-intersecting link removal and network disconnection problem of implicit planarization by presenting an explicit planarization algorithm called Localized Link Removal and Addition based Planarization Algorithm (LLRAP), which removes links only when it detects an intersection in its local neighborhood.

The localized link removal and addition based planarization (LLRAP) algorithm has two phases: 1) Local cross link detection and removal, which detects intersecting links locally and removes links to make the network planar. 2) Local

link addition, which adds links to ensure the connectivity of the network. Algorithm 7.1 summarizes the two phases.

7.4.1 Local cross link detection and removal

The local cross link detection phase detects all intersections in the local neighborhood. An intersection is detected, if any outgoing edge of a node intersects with the outgoing edges of its one hop neighbors. If such an intersection is found, one of the intersecting links is removed.



A simple intersection

Figure 7.5: A simple intersection

A node u executing *LocalCrossLinkDetectionAndRemoval* function of the algorithm 7.1, first makes a copy of the neighbors and then picks a neighbor v that has not been removed. It checks if the edge towards v, uv, intersects an edge wx (line 6), where w is another 1-hop neighbor of u and x is any 1-hop neighbor of w, as shown in Figure 7.5. If an intersection is detected, u then checks if the edge wv is present in the network. If wv is present, an alternate path from u to v, i.e. uwv, exists in the network. Hence, the edge uv is removed from the copy of neighbors. When all nodes perform this localized link removal operation, the resulting network becomes planar.

7.4.2 Local link addition

The second phase of the planarization algorithm is the local link addition phase. After the local cross link detection and removal phase, the network may get partitioned due to the localized removal process. Hence each node checks locally, if it could add any of the removed neighbors without causing any local intersections. For this, the node uses the original neighbor list and checks if a new link towards the removed neighbors causes any intersection in its current local neighborhood. This check is done from both end nodes of the new link and if no intersection is found the link is added.

Algorithm 7.1 LLRAP planarization algorithm

```
1: function LocalCrossLinkDetectionAndRemoval(Node u)
2: N(u) := origN(u)
3: for all v ∈ N(u) do
4:   for all w ∈ origN(u)\{v} do
5:     for all x ∈ origN(w)\{u, v} do
6:       if uv intersects wx then
7:         if v ∈ origN(w) then
8:           N(u) := N(u)\{v}
9:         end if
10:      end if
11:    end for
12:  end for
13: end for
14: end function

15: function LocalLinkAddition(Node u)
16: for all v ∈ origN(u)\N(u) do
17:   uv.markAddition = true
18:   for all w ∈ origN(u)\{v} do
19:     for all x ∈ N(w)\{u, v} do
20:       if uv intersects wx then
21:         uv.markAddition = false
22:       end if
23:     end for
24:   end for
25:   if uv.markAddition = true then
26:     if receive enableMsg from v then
27:       N(u) := N(u) ∪ {v}
28:       Instruct v to perform N(v) := N(v) ∪ {u}
29:     else
30:       send enableMsg to v
31:     end if
32:   end if
33: end for
34: end function
```

A node u executing *LocalLinkAddition* function of the LLRAP algorithm 7.1, checks if an edge uv towards its removed neighbor v intersects the edge wx, where w is any original 1-hop neighbor of u and x is any current 1-hop neighbor of w. If an intersection is found, it sets the marking of the edge uv for addition to be false. After checking all 1-hop neighbors of u and still the marking for addition remains true, u now checks if it has already received $enableMsg$ from v. The $enableMsg$ is a message which indicates that the sender node has finished the intersection check and wishes to add the new link. If u has not yet received this message from v, it sends the $enableMsg$ to v. On the other hand, if it has received this message, it adds v to its current neighbor list and instructs v, to add u to v's current neighbor list.

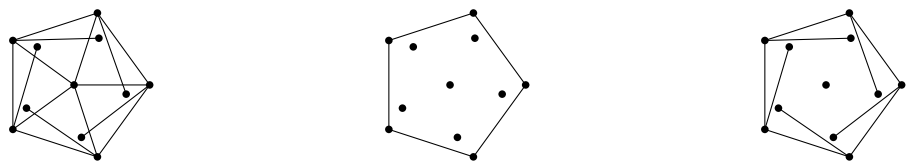
7.5 Theoretical Analysis

7.5.1 Graph properties

Our algorithm assumes one key property called *redundancy property* [146] for the input graphs to planarize them correctly.

Definition 1. *A graph satisfying redundancy property has: for any two intersecting edges, at least one node of the intersecting edges is directly connected to the remaining three nodes of the intersecting edges.*

When they satisfy the redundancy property, we claim that the local cross link detection detects intersections locally and the link removal process removes one of the intersecting edges and makes the network planar. We prove these claims in Section 7.5.2. However, the redundancy property alone is not sufficient to prove that our algorithm creates a connected planar graph for a connected input graph with redundancy property. This is illustrated in Figure 7.6, where the input graph which has redundancy property gets disconnected after LLRAP algorithm.



(a) Original graph with redundancy property

(b) After local cross link detection and removal

(c) After some steps of local link addition

Figure 7.6: Counter example

Hence, we introduce a new property called *coexistence property.*

Definition 2. *In a graph satisfying coexistence property, for any three existing edges uv, vw and wu in the graph, if there exists a node x lying inside the triangle $\triangle uvw$, then the edges ux, vx and wx also exist in the graph.*

We make a stronger assumption that our input graph satisfies coexistence property.

UDG is a subclass of graphs obeying these two properties. The redundancy property of UDG modeled networks is proved in [146]. We now show that UDG has *coexistence property.*

Lemma 1. *UDG modeled networks have coexistence property.*

Proof. A circle drawn with its center at one of the nodes of an existing triangle in the network and its radius as the maximum transmission radius of UDG, covers the entire area of the triangle. Hence any point inside the triangle is connected to the vertices of the triangle. □

7.5.2 Planarity

Lemma 2. *The local cross link detection and removal phase of LLRAP algorithm detects all intersecting edges with local 2-hop information and creates a planar network in graphs with redundancy property.*

Proof. Every intersecting edges has at least one redundant node joining the other three nodes of the intersecting edges due to the redundancy property. Let the intersecting edges be uv and wx as shown in figure 7.7 and the redundant node be w. When the *LocalCrossLinkDetectionAndRemoval* function is performed at the nodes u (or v), the intersection is locally detected with its 2-hop neighbor information and the node v (or u) is removed from its neighbor list's copy. Removing nodes from the copy of the neighbor list ensures that the node could still use all existing edges to detect and remove other intersections in its 2-hop neighborhood. Since all intersections are detected locally and one of the intersecting links is always removed, the network is planar after the local cross link detection and removal phase. □

Lemma 3. *The planar network graph remains planar after the local link addition phase of LLRAP algorithm in graphs with redundancy property.*

Proof. The local link addition phase adds a link that was in the original network graph but removed during the local cross link detection and removal phase, only if this link does not intersect any of the outgoing links from the one hop neighbors of both end nodes of the link which is to be added. Since all intersections are locally detectable with the 2-hop information according to lemma 2, the new probable intersection due to a link addition, is also locally detectable from one of the end nodes. Hence when a link is added without causing any local intersection detectable from both end nodes, this link does not cause nonplanarity and the planar network graph remains planar after the local link addition phase. □

7.5.3 Connectivity

Lemma 4. *The initially connected network remains connected after the local link addition phase of LLRAP algorithm in graphs with redundancy and coexistence properties.*

Proof. Consider a link uv, which was removed during *LocalCrossLinkDetectionAndRemoval* and which cannot be added during *LocalLinkAddition* from u, that causes network partition as depicted in Figure 7.7. The edge uv cannot be added during *LocalLinkAddition* only if this edge intersects another existing edge. Let wx be the existing edge that intersects uv and was kept during *LocalCrossLinkDetectionAndRemoval* at u which hinders uv addition. If wx is kept during *LocalCrossLinkDetectionAndRemoval* at u, from algorithm 7.1 line 7, it is evident that the edges wv and uw exist at that point of time.

Our assumption that the network is disconnected becomes true occurs only when no alternate path connecting u and v exists in the network. Let us consider all nodes lying inside the triangle $\triangle uwv$ and the corresponding alternate paths connecting u and v. Let $uw'v$ be the shortest path among them. If there is no such node in the uwv, then $w' = w$. The links uw' and $w'v$ exist according to the coexistence property.

However, if uw', $w'v$ or both uw' and $w'v$ are removed later during other *LocalCrossLinkDetectionAndRemoval* operations the path $uw'v$ does not exist in the network. In these cases if uw' or $w'v$ has been removed, it would be for the same reason uv was removed for: i.e. due to any link $w''x''$ intersecting with uw' or $w'v$, respectively.

Let $u_0 = u$, $v_0 = v$, $w_0 = w$, $w'_0 = w'$ and $x_0 = x$ as shown in Figure 7.7. If uw' was removed, we set $u_1 = u$ and $v_1 = w'$, else if $w'v$ was removed, we set $u_1 = w'$

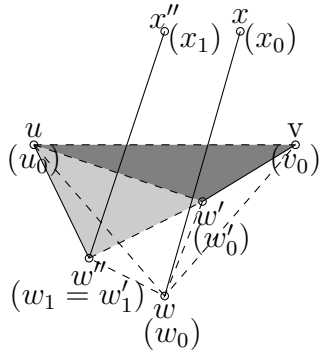


Figure 7.7: Intersecting links and alternate paths in networks with redundancy and coexistence properties

and $v_1 = v$. We also set $w_1 = w''$ and w'_1 to the node that connects u_1 and v_1 that has the shortest alternate path and lying inside the triangle $u_1w_1v_1$. In the same way we find the vertices u_1, v_1, w_1 and w'_1 from u_0, v_0, w_0 and w'_0, we recursively find the vertices u_i, v_i, w_i and w'_i from u_{i-1}, v_{i-1}, w_{i-1} and w'_{i-1}.

To prove connectivity, it is sufficient to show that no cycle occurs in this recursion, i.e. $u_iw'_i, w'_iv_i \notin \{u_jv_j : j < i\}$. In words, the two links needed to connect the end points of a removed link should not be the links removed before. As $u_iw'_iv_i$ is the shortest alternating path connecting u_iv_i inside the triangle $\triangle u_iw_iv_i$, $\forall i > j$ w'_i lie outside the triangle $u_jw'_jv_j$. This means that in each step of the recursion w'_i moves to a new distinct location that lies outside the region A formed by the union of previous triangles, i.e. $\triangle u_jw_jv_j$, $\forall j < i$. Moreover, the locus of w'_i always lies on one side of a line passing through uv and limited to the region where it does not intersect any $w_jx_j \forall j < i$, as the network considered during the link addition phase is planar.

As w'_i which becomes u_{i+1} or v_{i+1} in the next recursion step is never the same u_j or v_j, for all $j < i$, no cycle occurs in this recursion. Therefore, the network is connected after this phase. □

Theorem 1. *The LLRAP algorithm produces a connected planar graph in a connected graph with redundancy and coexistence properties.*

Proof. Lemma 2 and Lemma 3 prove that the network is planar when LLRAP algorithm is applied in graphs with redundancy property. In Lemma 4, we prove that the local link addition phase of LLRAP algorithm ensures the connectivity of planar graph created by the local cross link detection and removal phase in connected graphs with redundancy and coexistence properties. Hence the LLRAP

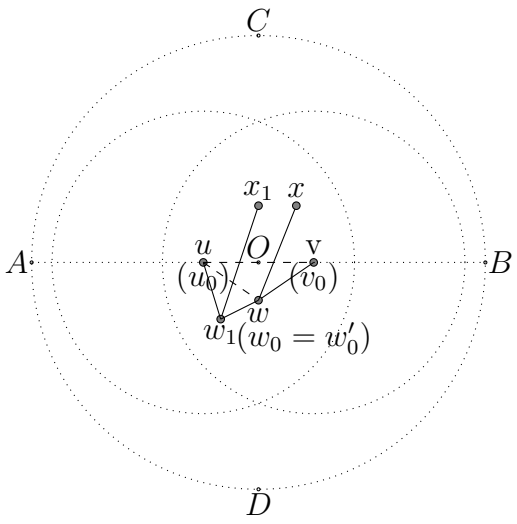


Figure 7.8: Illustration of weak spanner property

algorithm produces connected planar graphs for connected graphs with redundancy and coexistence properties. □

7.5.4 Weak spanner

Theorem 2. *The planar graph created by the LLRAP algorithm is a weak spanner in UDG modeled networks.*

Proof. For proving the weak spanner property, we follow the same arguments used to prove Lemma 4. We consider an edge uv, which was removed during *LocalCrossLinkDetectionAndRemoval* and cannot be added due to an existing link wx. If the links uw or wv is removed afterwards, an alternate path connecting the end nodes of the removed links exists in the network as the network is connected according to Lemma 4. This path lies inside the triangle $\triangle uwv$ (including the border path uwv) due to the coexistence property, unless there is another w_1x_1 with w_1 located on one side of a line passing through through u and v and outside the triangle $\triangle uwv$, and x_1 on the opposite side of the line, as shown in Figure 7.8.

The link w_ix_i cannot intersect any w_jx_j for all $j < i$, as the network considered during the link addition phase is planar and it has to intersect the edge u_iv_i. $\forall j < i$, w_i lie outside the triangle $u_jw_jv_j$. Hence all w_ix_i intersect the uv line. Since $\forall i$, $|w_ix_i| \leq R$ in a UDG modeled network, if we consider a circle whose center is at the middle of the line segment joining u and v and with radius $1.5R$, all u_i, v_j, w_i, and x_i lie inside this circle. This proves that the planar graph created by the 2-hop planarization algorithm is a weak spanner with $c = 1.5$. □

7.6 Empirical Analysis

Theoretical analysis shows that planar graphs created by LLRAP algorithm are weak c-spanners at least in UDG modeled networks. We now perform an empirical analysis on the spanning ratio of the planar graphs created by LLRAP algorithm and localized RNG, GG, and DT algorithms in UDG modeled networks.

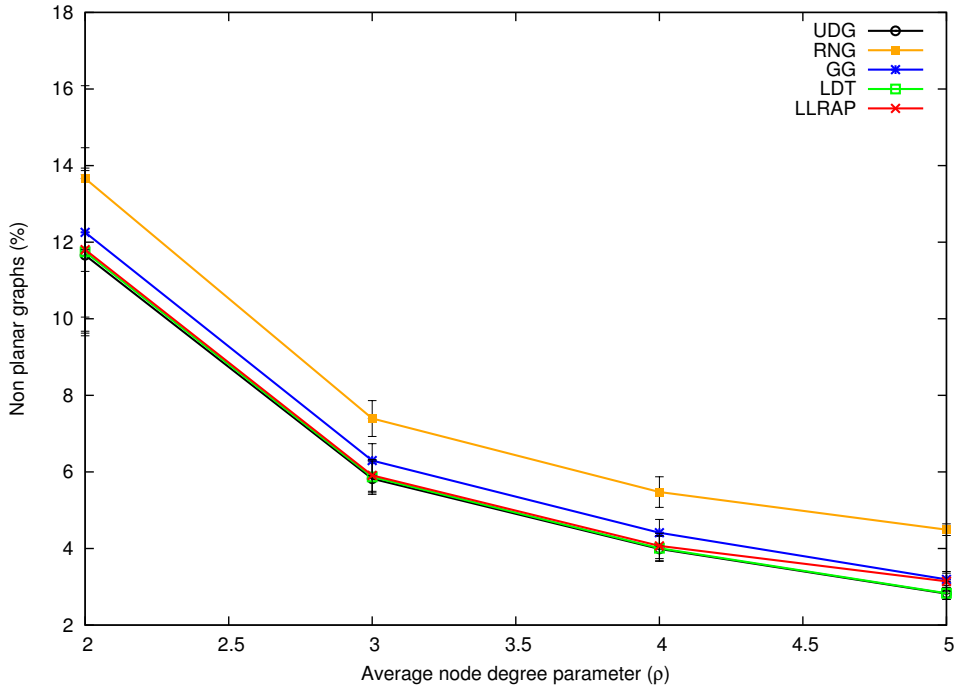


Figure 7.9: Spanning ratio of planarization algorithms

We create different networks in Shox network simulator by varying the *average node degree*, D, defined as:

$$D = \frac{\pi r^2}{A} * N, \tag{7.1}$$

where r is the transmission radius, A is the area of the field, and N is the total number of nodes in the network, keeping the area constant. Nodes are distributed randomly with a uniform distribution in this area. For each specific node degree parameter $\rho = \frac{D}{\pi}$, 100 connected networks are created. We set the UDG radius to 50 units.

Figure 7.9 shows the average spanning ratio of 100 networks at a confidence level of 95%, created in a field of size 500×500, when ρ is varied from 2 to 5. It shows that the spanning ratio of the planarized networks using Localized Delaunay Triangulation (LDT) and LLRAP algorithms are really close to the spanning ratio of the original UDG modeled networks. RNG and GG planarized networks have

the largest and second largest deviations respectively from the original values.

7.7 Summary

In this chapter, we proposed a new explicit localized graph planarization algorithm LLRAP that detects intersections locally and planarizes networks by removing them. Each node collects 2-hop neighbor information to detect local intersections and removes an edge that causes intersection, if it has a redundant path to the other node of the edge. A link addition phase ensures that the network remains connected after the planarization phase. LLRAP provably planarizes graphs without causing disconnection if the graph satisfies redundancy property and a new property introduced by us called coexistence property. Graphs satisfying these two properties are more generic than UDG. Theoretical analysis shows that planar graphs created by our algorithm are at least weak spanners in UDG modeled networks. Empirical analysis shows that LLRAP algorithm is as good as the best state of the art localized planarization algorithm, LDT.

CHAPTER 8

Geographic Routing in Real Wireless Networks

You can avoid reality, but you cannot avoid the consequences of avoiding reality.

Ayn Rand (1905-1982)

We have seen in Chapter 6 that among the various routing protocols for ad-hoc networks geographic routing is an attractive solution that could support self-optimized routing features. Geographic routing protocol consists of three main components [155]: (a) Greedy routing, (b) Planarization, and (c) FACE routing. Greedy routing if successfully delivers a packet finds routing paths comparable to the shortest possible path, but it alone does not guarantee the delivery of packets because of void regions. FACE routing on a plane graph does guarantee the delivery of packets, but often has paths longer than the shortest path. For improved performance, FACE routing is combined with greedy routing and is used as a void handling solution to overcome dead-ends when greedy routing fails. To make the network graph planar, localized planarization algorithms are used.

Existing geographic routing protocols discussed in Chapter 6 are not directly applicable for AMRoNets because they often make too idealistic assumptions on network graph properties or node location accuracy which do not hold in reality. For practical wireless network protocol designs, the performance of the protocols in wireless networks with

- Irregular radio range and

- Imprecise node location information

are very important to ensure successful operation in real AMRoNet routing scenarios. In this chapter we look at how these two factors adversely affect the performance of the existing geographic routing protocols discussed in Section 6.2 and the various solutions proposed to overcome them.

8.1 Real wireless networks

The main assumptions commonly used in the geographic routing are: (a) nodes detect their accurate locations and announce them (except in beacon-less routing), (b) changes in the topology are slow compared to the announcements and all nodes have a consistent view of the network, (c) radiation patterns of all nodes are exact and symmetric, and (d) there are no obstacles and hence nodes within the radio range can always communicate. However, in reality obstacles do exist, radiation patterns have irregular shapes, location measurements are often noisy and erroneous and nodes do not have a consistent view of the network. Violation of the first two factors leads to location inaccuracies and the last two to radio range irregularities.

8.1.1 Irregular radio range

Radiated pattern of nodes are often assumed to be perfect circles. Even for omni-directional antennas, this assumption does not hold [183]. Wireless channel characteristics (fading) and obstacles' presence make it highly irregular.

In literature, the evaluations of geographic routing protocols are often carried on an ideal network connectivity graph based on the Unit Disk Graph (UDG) model described in Section 7.1. UDGs are simple enough to derive strong theoretical results on cost and guarantee of routing protocols [110]. However, it is far from reality due to radio range irregularity.

A more general model considered in the literature is the Quasi (Unit) Disk Graph [110]. In a Quasi Disk Graph, two nodes are connected by an edge if their distance is less than or equal to a shorter radio range r and no edge exists between the nodes if it is greater than a longer range R as depicted in Figure 8.1. In the range between r and R the existence of an edge is not specified. The special case where $R = 1$ is called the Quasi Unit Disk Graph (QUDG or d-QUDG), where $d = \frac{r}{R}$ and has a value between 0 and 1.

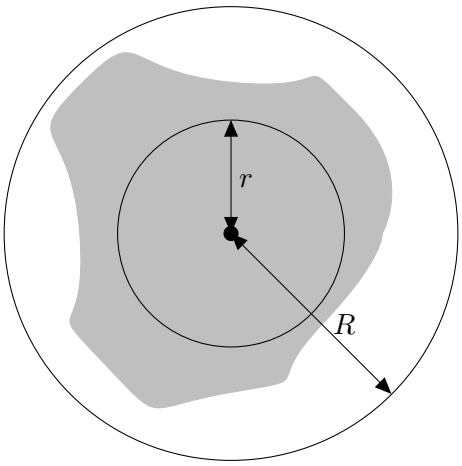


Figure 8.1: Quasi (Unit) Disk Graph

A more realistic model is the Log Normal Shadowing (LNS) [149]. It takes the presence of obstacles into account in the model. The path loss, the ratio of radiated power to the received power $\frac{P_{tx}}{P_{rcvd}(d)}$, at a distance d expressed in decibel is:

$$PL(d)[dB] = PL(d_0)[dB] + 10\gamma \log_{10} \frac{d}{d_0} + X_\sigma[dB], \tag{8.1}$$

where γ is the path loss exponent, X_σ is a zero-mean Gaussian random variable with variance σ^2, and $PL(d_0)$ is the reference path loss at a reference distance d_0. The random component X_σ is added to the signal attenuation to reflect the signal strength variations caused by shadowing.

The Received Signal Strength Indicator ($RSSI$) at a distance d, for a given transmitter output power $P_{tx}[dBm]$, is:

$$RSSI(d)[dBm] = P_{tx}[dBm] - PL(d)[dB] \tag{8.2}$$

Two nodes separated at a distance d are considered as neighbors when the $RSSI(d)[dBm]$ is greater than the receiver's sensitivity, RxS.

Given a certain distance d, we need to find out whether the probability of the received power $RSSI(d)$ is below the threshold RxS. Since only the path loss PL(d) is random, the probability can be expressed as a probability involving path loss, i.e. $P(RSSI(d) < RxS) = P(PL(d) > \beta)$, where β is maximum tolerable path loss. As $PL(d)$ is a Gaussian random variable and for any Gaussian random variable X with mean α and standard deviation σ, $P(X > \beta)$ can be expressed

as:

$$P(X > \beta) = Q\left(\frac{b-\alpha}{\sigma}\right) \tag{8.3}$$

where Q is the error function which is defined as $Q(z) = \frac{1}{\sqrt{2\pi}} \int_z^{+\infty} exp\left(-\frac{x^2}{2}\right) \mathrm{d}x$. Q can be calculated using numerical integration or look up tables. Figure 8.2 plots

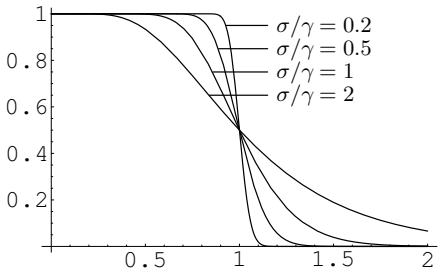


Figure 8.2: Connection probability versus the node distance for log-normal shadowing [131]

the connection probability for log-normal shadowing with node distance. The function is shown for different values of the shadowing deviation σ normalized to the path loss exponent γ, as the shape depends only on this ratio. For small ratios, the connection function becomes a step function and the resulting network graph a unit disk graph [131].

8.1.2 Localization errors

Geographic routing protocols assume that exact node position information is at hand. However, in practice, geographic location information is not exact; because of the unavailability or limitation of resources or because of errors in the localization techniques. In [102], location errors are classified into four metrics: a) absolute location inaccuracy, b) relative distance inaccuracy, c) absolute location inconsistency and d) relative distance inconsistency. Absolute location inaccuracy is the difference between the true location and the estimated location. Relative distance inaccuracy is the difference in relative distances. Absolute location inconsistency represents the difference of the same target node locations perceived by two nodes. Relative distance inconsistency is the difference in relative distance inaccuracy perceived by two nodes. In all localization systems, some of these errors are inevitable and magnitude of the error depends on the localization system type used, its implementation and the environment in which it is used [84]. A localization error of 1 - 10% of the radio range is very common in the best exist-

ing localization schemes [155]. Hence we need to take into account node location errors in the protocol designs.

To model location errors the most commonly considered error model is the Gaussian error model [155]. The mean value is usually set to 0 and standard deviation σ_{err} is a variable. The σ_{err} is often varied between 0 and 20% of the *radio range*. Radio range of UDG model is the radius R. For QUDG, radio range could be approximated to mean of the two radii, i.e. $\frac{r+R}{2}$. For LNS model, the radio range can be calculated using the equation 8.4, assuming $\sigma = 0$.

$$radio\ range = 10^{(P_t - PL(d_0) - RxS)/(10 * gamma))} \tag{8.4}$$

8.2 Geographic routing in real wireless networks

We now look at the impact of radio range irregularity and location errors on the performance of greedy and FACE routing.

8.2.1 Greedy Routing

8.2.1.1 Link reliability

In real wireless networks, wireless links can be extremely unreliable and [183] demonstrated the existence of a large *transitional region* where link quality has high variance. If greedy routing based on maximum advance is chosen in such cases, it might take a few transmission attempts to successfully forward a packet to a node in the transitional region with low link quality. Another node with a smaller advance could be reached on the first attempt. Hence, the cost for re-transmissions should be considered by the forwarding strategies. The cost for re-transmissions can be estimated by the number of transmissions needed to send a data packet over a specific link [152]. Thus by comparing the cost and advance, a balance between costly retransmissions over longer links and the increased hop count cost over shorter links can be made.

The expected number of re-transmissions can be derived from the packet reception rate (PRR) [152] which is defines as the ratio of successfully received packets over the total number of transmitted packets over a specific time period. It can serve as a reception probability metric for future transmissions. Choosing the neighbor with the highest value for the product of packet reception rate and the advance is considered in [156]. This criterion is an optimal metric for making

localized geographic forwarding decisions in lossy wireless networks with Automatic Repeat reQuest (ARQ) mechanisms and is also a good metric for No-ARQ scenarios [156]. Nodes using this criterion avoid selecting lossy links.

Another metric called Normalized Advance (NADV) [115] uses the advance achieved per unit link cost as the forwarding criterion and chooses the neighbor with largest NADV as the next hop node. Various types of link costs such as packet error rate, link delay, and energy consumption can be included in NADV. It tries to find the best trade-off between link cost and geographic proximity. Another similar metrics proposed is the cost-over-progress ratio presented in [111], which finds the minimal cost for transmission per progress towards the destination. Here the cost metric is the expected hop count based on the packet reception probability including acknowledgements, and the progress is considered as the advance towards the destination. Considering such cost metric significantly increases the packet delivery rate.

Greedy routing algorithms considering link reliability information such as the packet reception rate or packet error rate has to access the MAC layer for obtaining such information. Such greedy routing protocols are actually cross-layer protocols, as routing being a task of the network layer cannot utilize the MAC layer information directly in a traditional layered protocol design [34].

8.2.1.2 Location errors

The effect of localization errors on the performance of greedy routing is studied in [80]. They found that routing performance is not significantly affected when the error is less than 40% of the radio range. Location errors are not always bad as the packet drop in greedy routing due to voids (with precise location) may be avoided if location inaccuracy leads to a valid greedy forwarding path. However, the conditions for location inaccuracy to have good impact on geographic routing are very stringent and simulation studies in [102] shows that the bad effects such as increased packet drop and path length, outweighed the good effects.

In [102], an extensive simulation based analysis of the impact of location errors such as absolute and relative location inaccuracy and absolute and relative location inconsistency on the performance of greedy routing has been conducted. The authors reported increased packet drop rate with increase in location errors. The main reason for packet drop is the false void nodes encountered due to location errors. Most of the packet drop occurs within the destination range. This is due to the location inconsistency, where the last hop forwarding node perceives the

destination at a different location than the location perceived by the source node. Hence, the packet gets dropped at this void node. A small percentage (less than 2%) of packet drop was due to loops (single-hop and multi-hop) occurred due to relative distance inconsistency. Another side effect of location error was increase in the path length.

8.2.2 FACE routing and planarization

We have seen in Section 8.2.1 that for unreliable wireless links considering the link cost during greedy routing mitigates the problem considerably. The effect of localization errors degrades the performance of greedy routing. However, most of the packet drop occurs due to false local minimum and in geographic routing where such failures occurred are recovered by using FACE routing. Failures in the FACE routing lead to *persistent* protocol failures. Hence they are more critical in geographic routing.

We have seen the working of FACE routing in Section 6.2.2. During the FACE routing, the packet keeps moving to closer faces until it reaches the face containing destination node. FACE routing failures occurs due to the following two reasons:

- A disconnection or
- A cross-link

This has already been illustrated in Figure 6.4.

On analyzing these two problems, it has been identified that these problems occur because of planarization failures [155]. A correct planarization algorithm should not disconnect the connected network graph and should remove all cross-links in the graph. Hence, FACE routing failures are attributed to planarization failures. Let us now look at the impact of irregularity in radio range and location errors on planarization. Figure 8.3 shows examples of failures in RNG planarization due to radio irregularity and location errors resulting in disconnection and cross-links [155]. In Figure 8.3a, due to the irregularity in radiation pattern or due to the presence of obstacle, the link wv is absent. In Figure 8.3b, due to location errors w is considered as the witness node located at w'. In both cases, executing RNG algorithm at u removes the link uv and causes disconnection (if v also removes vu following uv removal) or at least an asymmetric link vu. In Figure 8.3c, due to radio irregularity and in Figure 8.3d, due to location errors, u fails to remove uv link resulting in cross-links.

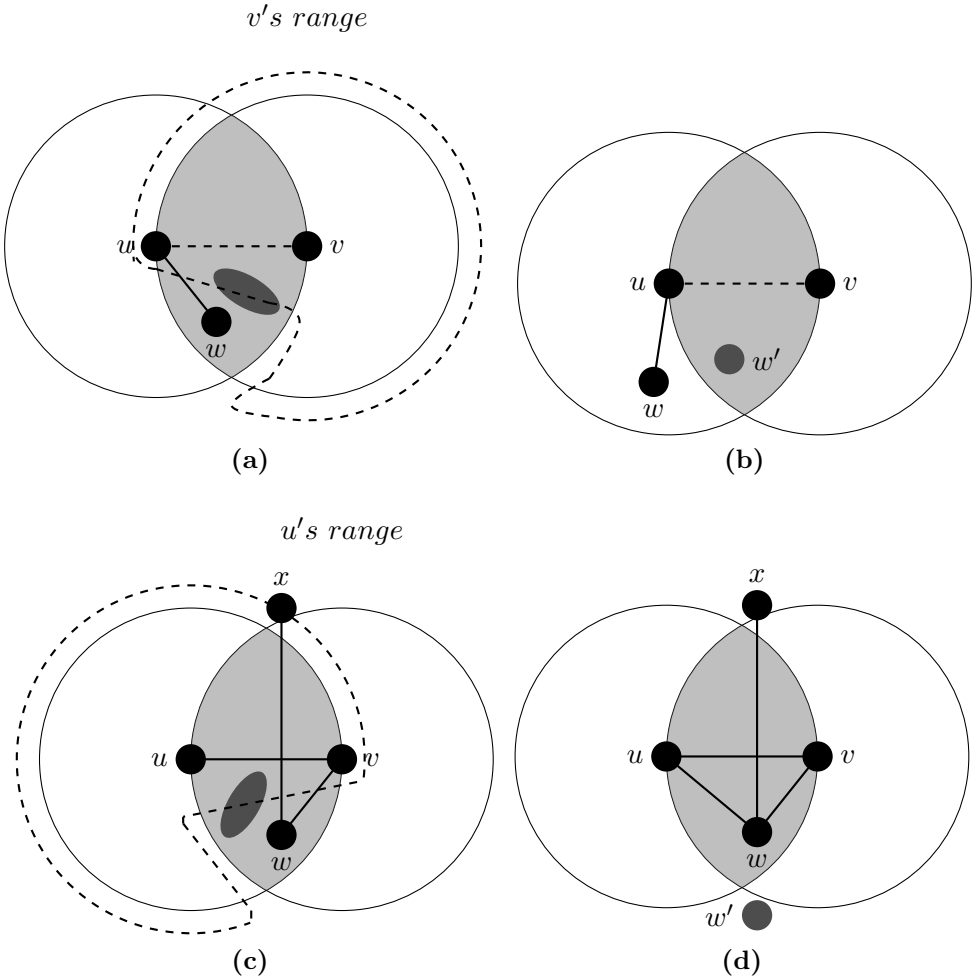


Figure 8.3: RNG planarization failures due to radio range irregularities and location errors

Localized planarization techniques such as GG, LDT, LMST and DP seen in Section 7.2 also fail to planarize the network correctly when radio irregularity or location errors are present. All these algorithms provably yield a connected planar graph only if the network graph obeys the UDG property. The explicit planarization algorithm LLRAP proposed by us planarizes graphs more general than UDG. However, arbitrary network graphs do not obey redundancy property or co-existence property. Example of violation of redundancy property due to radio range irregularity is shown in Figure 8.4. It has been proved in [155] for arbitrary connectivity graphs, it is not possible to avoid disconnections and cross-links at the same time. Figure 8.4b is an example that illustrate this fact, where the connected network graph gets disconnected on removing any of the cross-links uv or wx.

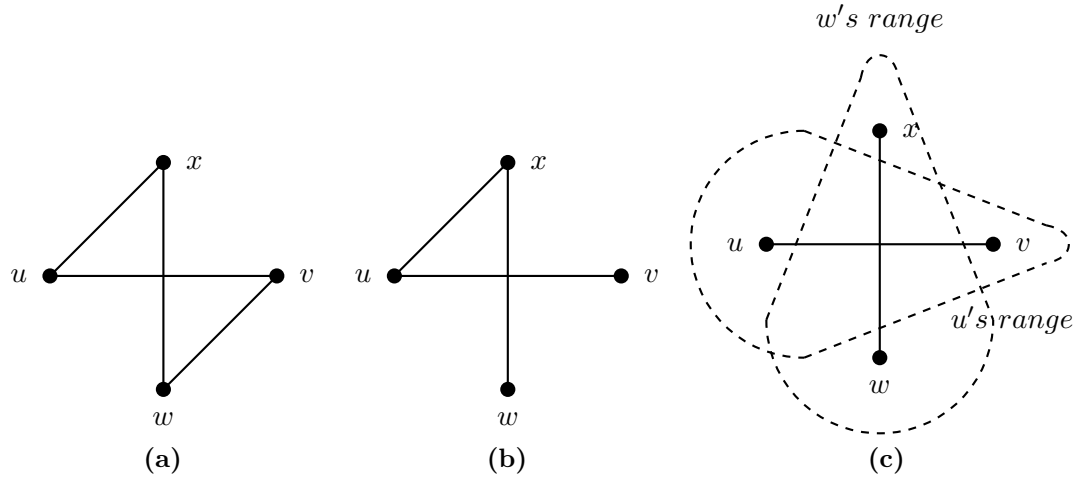


Figure 8.4: Intersections without redundancy property

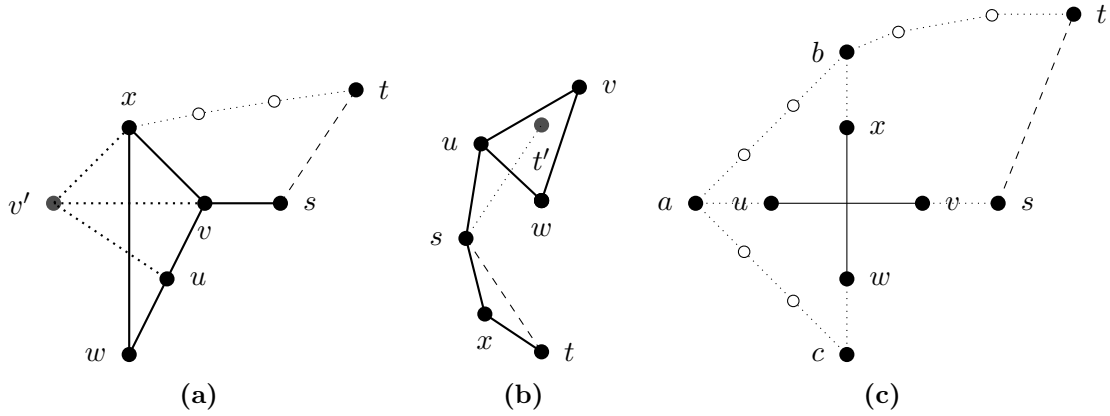


Figure 8.5: FACE routing failure due to radio range irregularities and location errors

Figure 8.5a shows an example where location error of a single node v introduces new cross-links wx intersecting sv' and uv'. In these intersections redundancy property is not satisfied. If there was no error, FACE routing from source s to destination t follows the path $s - v - x$ and reaches t. With the error, the packet from s take the path $s - v' - x - w - u - v' - s$ and gets dropped. Figure 8.5b shows another example of location error causing FACE routing failure.

We have seen that both irregularity in radio range and location error can cause disconnections and cross-links in the network which in turn causes FACE routing failure. Even small location errors can lead to incorrect routing decisions with severe performance degradation [102,155]. For graphs with arbitrary connectivity,

local algorithms that use only single-hop information or even a fixed number of hops cannot detect all cross-links in the graph and cannot guarantee delivery for FACE routing [155]. Any search of a fixed number of hops is not sufficient to guarantee the detection of all cross-links with arbitrary connectivity. This is illustrated in Figure 8.5c, where the cross links uv and wv are detected by searching the path $v - u - a - b - x - w - c$ which could be arbitrarily large and cannot be bounded.

8.2.2.1 Restricted wireless model

Let us now look at the planarization algorithms developed for network graphs which are more general than UDGs. The *mutual witness* approach presented in [155] addresses the network disconnection problem due to incorrect edge removal during GG or RNG planarization process. An edge is only removed, if both endpoints are connected to the witness node. Thus by considering a mutual witness node, the disconnection problem could be eliminated. However, the cross-link problem is not eliminated by this approach.

In Quasi Unit Disk Graphs computing a Gabriel graph may produce disconnections. However, after adding virtual edges in a d-QUDG with $d \geq \frac{1}{\sqrt{2}}$ and applying Gabriel Graph results in a planar connected graph with a mixture of real and virtual edges [20]. For an edge uv, any witness node w within the Gabriel circle is connected either to u or v if $d \geq \frac{1}{\sqrt{2}}$. Hence during the uv link removal due to w, it is possible to add a virtual link between w and u or v, if the link does not exist in the original graph. After planarizing the extended graph, FACE routing can be applied. Routing over virtual edges is then performed by communication between endpoints, which could be multiple hops in the original graph. The length of this routing path can be bounded if there is a minimum distance between any two nodes; otherwise, it is unbounded. The route length bound can be preserved without the minimum distance assumption as shown in [110] where a connected dominating set (CDS) is first extracted from the network graph and clustering is then used to reduce the number of edges of the CDS. In such graphs the planarization algorithm of Barrière et al. [20] is applied and geographic routing with asymptotically optimality could be achieved [110].

An alternate idea to planarize graphs is to use virtual nodes instead of the virtual links [122]. To obtain planarity, each edge intersection is replaced with a virtual node and a real node serves as a proxy for the virtual node. In d-QUDG with $d \geq \frac{1}{\sqrt{2}}$, these intersections can be detected locally and the algorithm

guarantees that an edge between a virtual node and a neighbor is realized by a constant-hop path in the real network. It works only if the intersections are detected locally, i.e. when $d \geq \frac{1}{\sqrt{2}}$.

Another approach presented in [63] selects a set of landmark nodes forming a k-hop independent set. Nodes are associated to these landmarks and the plane is partitioned into Voronoi tiles. From the Voronoi tiling, a Combinatorial Delaunay Map (CDM) is constructed which is planar for any d-QUDG with $d \geq \frac{1}{\sqrt{2}}$. The routing protocol uses the CDM as a macroscopic guide for routing; i.e. a node uses locations of nearby landmarks to determine to which neighboring Voronoi tile the packet needs to be forwarded. The packet is then forwarded to that tile using a gradient descent method.

8.2.2.2 Arbitrary network models

FACE routing protocol that works on arbitrary network graphs is the Cross-Link Detection Protocol (CLDP) [103]. In CLDP, each node probes its links to see if they are crossed by other links. A probe initially contains the locations of the endpoints of the link being probed, and traverses the graph using the right-hand rule. When the probe is sent along a face, each node checks whether the next link on the traversal intersects the edge specified in the probe message. If an intersection is detected, the link is marked non-routable unless the probing message has traversed this link twice in opposite directions. In such cases, marking a link as non-routable (i.e. removing a link) causes network partition. The set of routable links forms a safe routable subgraph where FACE routing is guaranteed not to fail. Greedy routing uses the full graph, i.e. it includes the links marked non-routable by CLDP. Only the FACE routing uses the CLDP-derived routable subgraph to recover from void nodes.

When there are multiple crosslinks, repeated probing is required. Concurrent probing will lead to disconnections, if two crossing links are marked non-routable at the same time. Locking schemes and tie breaking rules are needed to avoid such situations. CLDP always produces a safe routable subgraph from any arbitrary input connected graph. As the probe messages cause significant overhead in the network, an alternative approach called Lazy Cross-Link Removal [75] that does not proactively remove the cross links, but applies CLDP when loops are found has been proposed.

8.3 Summary

We have seen in this chapter that irregular radio range and imprecise node location information are the two main challenges for geographic routing protocols in real wireless networks. We considered two models, QUDG and LNS, that could model radio irregularities better than UDG models. We also analyzed how to model location errors. We looked at the various greedy routing protocols that consider link reliability in the forwarding decisions and the impact of location errors on greedy routing. Location errors degrade the performance of greedy routing. However, most of the packet drops occur due to false local minimum and in geographic routing such failures are recovered by using FACE routing. FACE routing fails when there are disconnections or cross-links in the network graph which lead to persistent protocol failures. These problems are due incorrect planarization process which removes incorrect edges causing network partitions and unidirectional links or fail to remove edges causing intersecting links. Hence planarization in real wireless network is very important for geographic routing protocols. We also looked at some of the localized planarization approaches for d-QUDG with $d \geq \frac{1}{\sqrt{2}}$ and nonlocal approaches for arbitrary network graphs[1].

[1]They do not create planar graphs but a safe routable subgraph for FACE routing

CHAPTER 9

Graph Planarization in Realistic Wireless Networks

We have seen in Chapter 8 that irregular radio ranges and location errors degrade the performance of geographic routing protocol. FACE routing failures attribute to most of the geographic routing failures and leads to persistent protocol failures. FACE routing failures arise from incorrect planarization which removes incorrect edges causing network partitions and unidirectional links or fail to remove edges causing intersecting links.

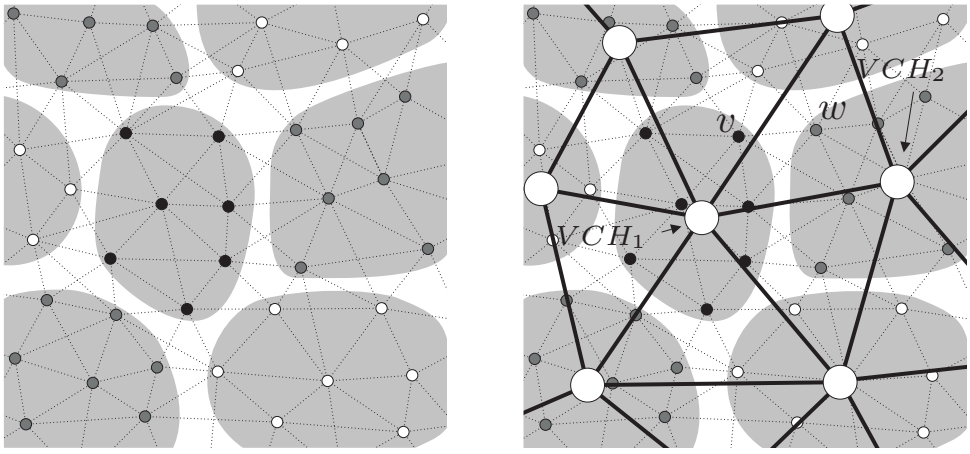
Localized planarization algorithms are highly preferred for geographic routing to preserve its locality characteristic, as it is the base for its scalability and efficiency. Existing localized planarization techniques discussed in Section 7.2 work well for restricted wireless models such as UDG or d-QUDG with $d \geq \frac{1}{\sqrt{2}}$. They do not work correctly in realistic wireless networks, which in most cases do not obey these restrictions. The planarization algorithm LLRAP works in graphs more general than UDG, but it also fails in arbitrary network graphs. In addition, location inaccuracies degrade the performance of localized planarization algorithms. Even small errors can lead to incorrect routing decisions with noticeable performance degradation [102, 155].

The planarization protocols [75, 103] designed for arbitrary networks are non-local. The probe message in these protocols may have to travel more than constant hops between the nodes to detect intersections. Thus, we have the following dilemma: there are topology control schemes which transform arbitrary 2D graphs (including localization faults) into a topology where FACE routing is always suc-

cessful. However, these approaches are not localized. On the other hand, all known localized approaches cannot be correctly applied on arbitrary 2D graphs and do not work when location errors are present. In this paper, we solve this dilemma utilizing a basic structural property of realistic wireless networks; i.e. the links cannot be arbitrarily long. We describe a localized planarization algorithm for realistic wireless networks which is location fault tolerant and produces planar graphs in most cases, using this property.

9.1 Topological Cluster-based planarization algorithm

The key idea of our approach is that, at a local scale even though the network graph might contain many intersecting links which are not detected by local algorithms, at a larger scale those intersections disappear or easily detectable by local algorithms. This is based on the observation that Euclidean edge lengths in ad-hoc wireless network graphs cannot be arbitrarily long, as seen in Figure 9.1a. Thus when we aggregate nodes into clusters, the limited edge length assures that the cluster interconnections do not intersect in most cases.



(a) Clustering of nodes **(b)** An overlay graph construction

Figure 9.1: The general idea of the planarization algorithm

Our planarization algorithm starts with topology-based clustering to aggregate nodes into clusters. The topology-based clustering is a distributed process which uses only local information exchange. Clustering creates a logical hierarchy in the network with dedicated cluster heads handling all inter-cluster routing decisions.

An *overlay graph* is constructed with the cluster heads constituting its vertices. Links joining the neighboring cluster heads constitute the edges in the overlay graph. Figure 9.1b shows an overlay graph created by topological clustering for illustration.

The overlay graphs may contain a few intersections and may not be planar. Hence we apply planarization algorithms at the overlay level to make it planar. The details of the planarization process stages are described next.

9.1.1 Clustering

The first stage of the planarization algorithm is cluster formation. In this stage the nodes are grouped into clusters based on any multi-hop clustering algorithm. We have developed a clustering algorithm called *k-hop clustering* based on the CONID (connectivity ID) [35], where k is the cluster depth or hop count. Our algorithm differs from [35] in the selection process of cluster head and cluster member nodes. This makes it more robust against node location errors, as we see later in the analysis. The k-hop clustering algorithm creates clusters with cluster heads that are separated at least k-hops apart. It uses connectivity as the primary key in clustering decisions, where connectivity is defined as:

Definition 3. *The connectivity of a node u,* $Con(u) = |S_u|$*, where* $S_u = \{v_1, v_2, \ldots\}$ *such that* $\forall$ *nodes* $v_i \in S_u$*, a path* $|p(u, \ldots, vi)| \leq k$ *from u to* v_i *exists and nodes along* $|p(u, \ldots, vi)|$ *do not belong to another cluster.*

For k = 1, the connectivity is equivalent to the node degree.

The k-hop clustering algorithm works as follows. All nodes periodically check their status to determine whether they are clustered or not as given in the procedure *cluster-all-node* in Algorithm 9.1. If there exists any node which does not belong to any cluster, it is triggered to create a new cluster or join an existing cluster as given in the procedure *trigger-cluster-node* in Algorithm 9.1. If the triggered node is an isolated node, i.e. without any neighbors, it changes its status to *clustered* and creates a new cluster with the current node as its only member node. All other triggered nodes create a k-hop neighbor list S and calculate their connectivity as given in definition 1.

The node with higher connectivity is preferred to nodes with lower ones. The nodes which have the highest connectivity among their k-hop neighbors become *winner nodes*. If two nodes u and v have the same connectivity and their connectivities are maximum in the set $S_u \cup \{u\}$ and $S_v \cup \{v\}$ respectively, then their

Algorithm 9.1 k-hop Clustering

1: **procedure** cluster-all-nodes
2: **while** $\forall$ *Node* $n \in V : n.status != CLUSTERED$ **do**
3: **if** $neighbors(n) = \emptyset$ **then**
4: $node.status = CLUSTERED$
5: **else**
6: trigger-cluster-node (n)
7: **end if**
8: **end while**
9: **end procedure**

10: **procedure** trigger-cluster-node (Node n)
11: Determine k-hop neighbor set, S
12: Calculate the connectivity, $Con(n)$
13: **if** Winner(n, S) = $true$ **then**
14: Cluster, $C \leftarrow \emptyset$
15: $C' = \bigcup_{i \in neighbors(n)} C(i)$
16: **if** $Con(n) < Threshold$ & $C' != \emptyset$ **then**
17: $C = Mode(C') \cup \{n\}$
18: **else**
19: $C = \{n\}$
20: $n.state = CLUSTER\_HEAD$
21: **for all** *node* $v_i \in S$ **do**
22: $C = C \cup \{v_i\}$
23: $v_i.status = CLUSTERED$
24: **end for**
25: **end if**
26: **end if**
27: **end procedure**

28: **procedure** Winner (Node n, Set S)
29: **for all** *Node* $v_i \in S$ **do**
30: **if** $Con(n) < Con(v_i)$ **then**
31: **return** $false$
32: **else if** $Con(n) = Con(v_i)$ & $ID(v_i) > ID(n)$ **then**
33: **return** $false$
34: **end if**
35: **end for**
36: **return** $true$
37: **end procedure**

node ids are compared to make the decision (see the procedure *Winner* in Algorithm 9.1). The one with the lower id then wins.

Prior to the cluster creation, a check is done to determine whether the connectivity of the node is above a threshold size, which is usually k. If connectivity is not above the threshold size, the node finds a cluster in which the majority of its neighbors are members, and joins that cluster. The node then requests all nodes in S to join this cluster. This *connectivity-threshold* step prevents the formation of clusters with very few members in the network. Though prevention of numerous small clusters is optional from the point of view of planarization, it helps to make the algorithm robust against node location errors. Due to the joining of new nodes, the cluster depths are increased at most by the threshold size. Usually it is increased only by a small fraction of the threshold size. Choosing small threshold values keeps this increase limited. In our experiments, the threshold is chosen to be the cluster depth k.

9.1.2 Overlay graph

The next process in the planarization algorithm is the overlay graph construction given in Algorithm 9.2. An overlay network could be constructed using the winner nodes which initiated the cluster creation; but to make the overlay vertices location fault tolerant, we create *virtual nodes* at the geometric center of the clusters. Virtual node averages out location errors in the cluster. The position of the virtual nodes is calculated as:

$$P_v(x, y) = \sum_{i=1..,n} \frac{P_i}{n}, \tag{9.1}$$

where n is the number of nodes in the cluster and P_i is the position of the i^{th} member of the cluster. These nodes constitute the virtual cluster heads (VCH) of the clusters. In practice, the winner node or a node which is close to the centroid of the cluster serves the functions of VCH. The node id of VCH is the actual id of the node which is serving it. The cluster is also represented by this id. After creating the VCH, the information about the VCH and the node serving it, is passed to all members of the cluster.

A virtual overlay graph $OG = (V', E')$ is created with the virtual cluster heads as its vertices. For a network graph $G = (V, E)$, the set of vertices of the overlay graph $V' = \bigcup_i \{VCH(v_i)\}$ for all $v_i \in V$, where $VCH(v_i)$ is the virtual cluster

Algorithm 9.2 Overlay graph

```
1: procedure Overlay (Cluster C_i)
2: Create virtual cluster head VCH
3: VCH.id = Winner.id
4: j=1
5: for all Node n_j ∈ C_i do
6:    Pos_VCH = (Pos_VCH*(j−1)+Pos_{n_j}) / j
7:    for all Node n ∈ neighbors(n_j) do
8:       if n.clusterId != VCH.id then
9:          neighbors(VCH).add(n.VCH)
10:      end if
11:   end for
12: end for
13: end procedure
```

head of node v_i. The set of edges of the overlay graph $E' = \bigcup\{e_i\}$, where an edge e_i exists between VCH_i of cluster C_i and VCH_j of cluster C_j, if $\exists v \in C_i$ and $w \in C_j$ such that, an edge $(v, w) \in E$ exists in G. For example, the virtual edge between VCH_1 and VCH_2 shown in Figure 9.1b exists, as there is a link between nodes v and w in the network graph.

For constructing the virtual links, each VCH sends queries to the boundary nodes of its cluster. If any boundary node has a link to another node with a different VCH, a virtual link is created between these cluster heads.

9.1.3 Overlay graph planarization

The last stage of the algorithm is the overlay graph planarization. Overlay graphs are mostly planar especially when the k values are large, but when k is small and network density is very high, they may not be planar. To make the overlay graphs planar, existing implicit planarization algorithms cannot be used as they do not obey UDG property. Implicit link removal based on geometric properties leads to disconnection as seen in Section 7.3. Explicit planarization algorithms are better choice in such scenarios. We choose $LLRAP$ algorithm described in Section 7.4 for planarizing overlay graphs.

The first phase of LLRAP algorithm, the local cross link detection and removal phase, detects all intersections in the local neighborhood. For cross link detection, all nodes $v'_i \in V'$ in the overlay graph $OG(V', E')$, collect their 2-hop neighbor information. The 2-hop neighbors of v'_i are those nodes which are at-most 2-hops

away from v_i', when traversed over the virtual links $e' \in E'$. An intersection is detected, if any of the outgoing edge of a node intersects with the outgoing edges of its one hop neighbors.

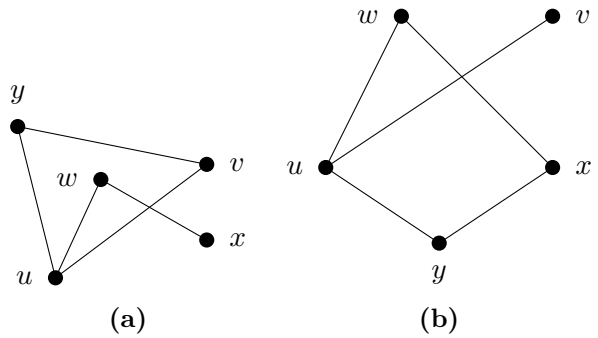


Figure 9.2: Overlay graphs without redundancy property

A node u executing local cross link detection and removal phase checks if the link uv intersects the link wx and if so, it removes the link uv when the link wv is present in the network. When the network graph satisfies the redundancy property, checking wv's presence works correctly. However, if we apply LLRAP at the overlay graphs that has no redundancy property, e.g. in graphs shown in Figure 9.2, overlay node u detects the intersection locally, but does not remove it as link wv is absent. Hence we modify the condition of link removal with the following rules:

- if there exists a path from u to v, other than the direct link uv, with path length at-most 2-hops, then remove the link uv
- else, if there exists a path from u to x, not through the node w, with path length at-most 2-hops, then instruct w to remove the link wx

The second phase of the planarization algorithm, the local link addition phase, adds those removed links whose additions do not cause any violation to planarity. This is done carefully with several message exchanges as described in Section 7.4.2. As the virtual links of the overlay graphs are realized with multi-hop paths, the local link addition phase incur considerable message overhead at the overlay level. Hence we propose a simple alternative which avoids the local link addition phase, but also takes care to prevent network partitions during local cross link detection and removal phase. We call this variant of local cross link detection and removal phase as *Cross Link Detection and Repair* (CLDR). The pseudo code of the algorithm is given in Algorithm 9.3. In CLDR, when a node removes an outgoing edge,

Algorithm 9.3 Cross Link Detection and Repair

```
1: procedure CLDR (OverlayNode u)
2: N(u) := origN(u)
3: for all v ∈ N(u) do
4:   for all w ∈ origN(u)\{v} do
5:     for all x ∈ N(w)\{u, v} do
6:       if uv intersects wx then
7:         if ∃y ∈ N(u)\{v} & uy, yv ∈ E then
8:           N(u) := N(u)\{v}
9:           Send message uv removed
10:        else if (∃y ∈ N(u)\{w} ∪ {u} & uy, yx ∈ E) then
11:          Instruct w to remove wx
12:        end if
13:      end if
14:    end for
15:  end for
16:  if receive message vx removed then
17:    N(u) := N(u)\{v}
18:    N(v) := N(v)\{x}
19:  end if
20: end for
21: end procedure
```

it informs all nodes in its neighborhood about the link removal. On receiving this notification, nodes update their 2-hop neighborhood list and ignore removed links during cross link detection checks. Concurrent link removals may cause network disconnection which could be avoided by using simple tie-break rules or locking methods [103].

9.2 Modeling and Simulation

9.2.1 Wireless Model

We evaluate our planarization algorithm using the Log Normal Shadowing Model (LNS) described in Section 8.1.1. The path loss exponent γ varies between 2 (free-space) and 6 (obstructed in-buildings) whereas, variance σ^2 varies between 3.7 and 12.8 [98, 157]. In indoor environments, for a frequency of 2.45 GHz, $\gamma = 2$ for line of sight (LOS) and $\gamma = 3.5$ for non-line of sight (NLOS) [114]. We choose $\gamma = 3.25$, slightly less than the NLOS value and $\sigma = 2.5$ for most of the experiments, but we

also vary them to study their effects on planarity. The reference path loss $PL(d_0)$, for a reference distance $d_0 = 1\ m$ varies between -50 and $-30\ dBm$ [98] and we set $PL(d_0)$ to $-40\ dBm$. The path loss can be calculated from equation 8.1 using these values.

The transmission power levels of IEEE 802.11 often vary between $-15\ dBm$ to 15 dBm [134]. We set the P_{tx} to 15 dBm. The receiver sensitivity varies with data transmission rates. E.g. IEEE 802.11g's RxS is $-88\ dBm$ at 6 $Mbps$ and $-66\ dBm$ at 54 $Mbps$ [134]. We set the RxS to $-80\ dBm$ for our experiments.

Besides LNS, we consider Unit Disk Graph (UDG) and Quasi Unit Disk Graph (d-QUDG) models as well in our simulations. In UDG, we set R to 50 units in our tests. For d-QUDG, we set d to 0.5, with $r = 25$ and $R = 50$ units. The performance of our algorithm at this d value, which is less than $\frac{1}{\sqrt{2}}$ is specially interesting, as existing localized planarization algorithms cannot guarantee planarity at such low values as discussed in Section 8.2.2.1.

9.2.2 Simulation setup

We create different networks in ShoX network simulator by varying the parameters like field size and *average node degree*, D, defined in equation 7.1. Now onwards we use a parameter $\rho = \frac{D}{\pi}$ to represent the node degree. In most of the experiments, we use the field size 500×500 square units, with N varying from 100 to 500. To test the scalability of the algorithm, we also consider a field size of 2500×2500 square units, with N varying from 2500 to 12500. The tests conducted on the former field are evaluated on 100 different network configurations while on the latter field, are evaluated only on 10 different network configurations due to the enormous time consumption of each test.

A graph planarity test is conducted to check the planarity of the overlay graph. It classifies the graphs into planar and nonplanar groups and reports the total number of intersecting links in nonplanar cases.

9.3 Performance Evaluation and Analysis

To analyze the performance of the planarization algorithm, we vary parameters such as hop count k used in k-hop clustering and node degree ρ. We record the percentage of nonplanar graphs in the LNS modeled networks. Table 9.1 shows the result of the planarization algorithm on two field sizes, 500×500 and 2500×2500

Field Size 500 × 500												
ρ	Hop Count											
	1		2		3		4		5		6	
	a	b	a	b	a	b	a	b	a	b	a	b
2	59	0	17	0	3	0	1	0	0	0	0	0
3	99	0	51	0	9	0	1	0	1	0	0	0
4	100	0	73	0	10	0	1	0	0	0	0	0
5	100	0	84	0	17	0	2	0	0	0	0	0
Field Size 2500 × 2500												
ρ	Hop Count											
	2		4		6		8		10		12	
	a	b	a	b	a	b	a	b	a	b	a	b
2	100	0	60	0	50	0	50	0	20	0	20	0
3	100	0	100	0	70	0	50	0	20	0	10	0
4	100	0	100	0	100	0	70	0	30	0	10	0
5	100	0	100	0	100	0	50	0	40	0	50	0

Table 9.1: Performance of the planarization algorithm

square units, with ρ varying from 2 to 5. Hop count k varies from 1 to 6 in the first field and 2 to 12 in the second field. The column $k.a$ indicates the percentage of nonplanar graphs before applying $CLDR$ step and the column $k.b$ indicates the percentage after applying $CLDR$.

Table 9.1 shows that the planarity of the graphs increases with the hop counts and decreases with the node degrees and most important, overlay graphs are always planar in all test cases when $CLDR$ is applied. Figure 9.3 shows the effect of hop count on planarity more clearly, where the percentage of nonplanar graphs without $CLDR$ is plotted at a confidence level of 95% for three different network models. The field size used in these experiments is 500 × 500 square units and the node degree (ρ) is 3. It shows that, for smaller k values we get fewer planar overlay graphs. With increasing k value, more graphs become planar. When k is increased to values comparable to 25% of the network diameter, most of the overlay graphs are planar.

The experiments conducted on the field size 2500×2500 square units also affirm that planarity increases with the hop count. At higher k values like 10 or 12, the majority of the test cases are planar. If we increase k further to values comparable to the network diameter, the probability of getting planar overlay graphs is very high as the total number of clusters in the network at large k values is very small. This decreases the probability of having intersecting links.

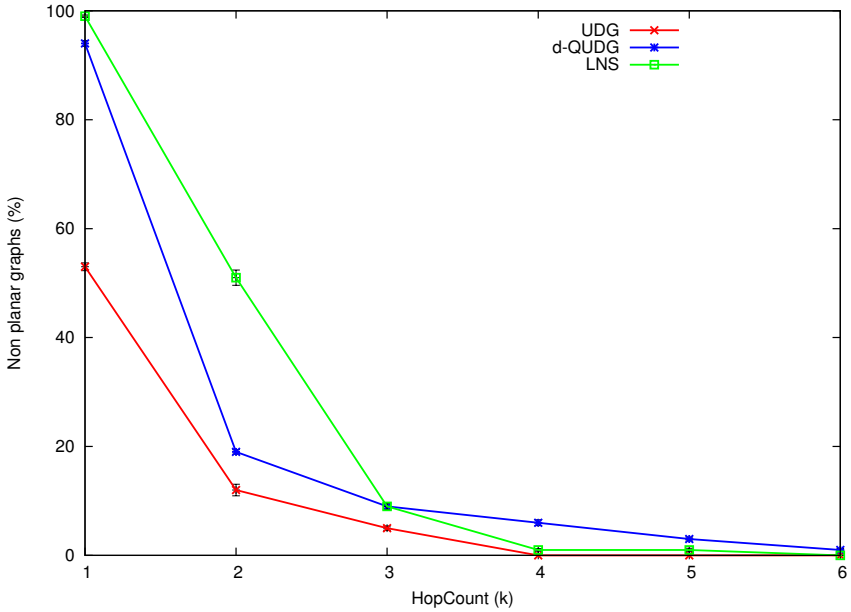


Figure 9.3: Effect of cluster depth on planarity

To obtain planar overlay graphs with low k values, the $CLDR$ step is needed. Columns b of table 9.1 shows that with the $CLDR$ step, all graphs are planar irrespective of the average node degree. It is also interesting to note that for small hop counts such as k=1, the overlay graphs are planar in all simulation runs. However, we may need $k > 1$ in certain situations, as discussed in Section 9.3.2 and Section 9.3.3.

9.3.1 Performance of existing planarization algorithms

We have seen in Section 7.2 graph algorithms such as RNG, GG and LDT create planar sub-graphs locally from the full network graph using distributed algorithms. These algorithms provably yield connected planar graphs in connected UDG networks. We now evaluate their performance in LNS and QUDG modeled networks.

Figure 9.4 shows the results of the experiments at a confidence level of 95% on the field of size 500×500 square units, when ρ is varied from 1 to 5. The figure shows that LDT, GG and RNG perform very badly in planarizing realistic wireless network graphs. For $\rho > 2$, less than 10% of the GG subgraphs are planar and for the same ρ value, none of the LDT subgraphs are planar. The planarization algorithm we proposed, always created planar graphs, irrespective of node degrees in the LNS and QUDG modeled networks.

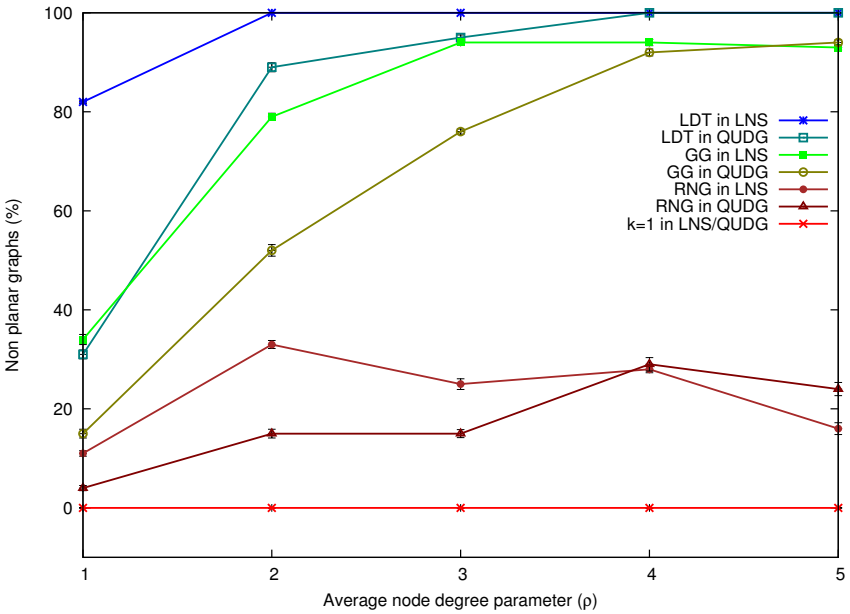


Figure 9.4: Comparison of nonplanar graphs in LNS and QUDG modeled networks

9.3.2 Effect of model parameters on planarity

The experiments discussed in the above section are based on fixed values for path loss exponent γ and variance σ^2 of LNS model. In this section, we analyze the effect of these parameters on the planarity of the graphs.

To study the effect of LNS variance, we vary σ fixing the γ value. Figure 9.5 shows the result of the planarization algorithms at a confidence level of 95% on field size 500×500 square units with $\rho \approx 3$ and $\gamma = 3.25$. Empirical studies show that σ^2 varies between 3.7 and 12.8 [98,157]. Hence, we vary σ from 0 to 4 (σ^2 from 0 to 16) in our experiments. The results of LDT, GG and RNG planarization algorithms show that the variance has significant impact on the planarity of the graphs. Planarity decreases with the increase in variance. As variance increases, nodes that are closer may not be connected any more, but those far apart may get connected. Such link irregularities lead to intersections which cannot be repaired without causing disconnection.

The results of our planarization algorithm with $CLDR$, show that the proposed algorithm is robust against the σ variations. The $k = 1$ planarization algorithm has a few nonplanar cases, but only at σ values outside the normal range. The $k = 2$ planarization algorithm always creates planar graphs even for extreme variances. This is because larger k values subside local link irregularities better.

In a study on the effect of path loss exponent on planarity by varying γ fixing

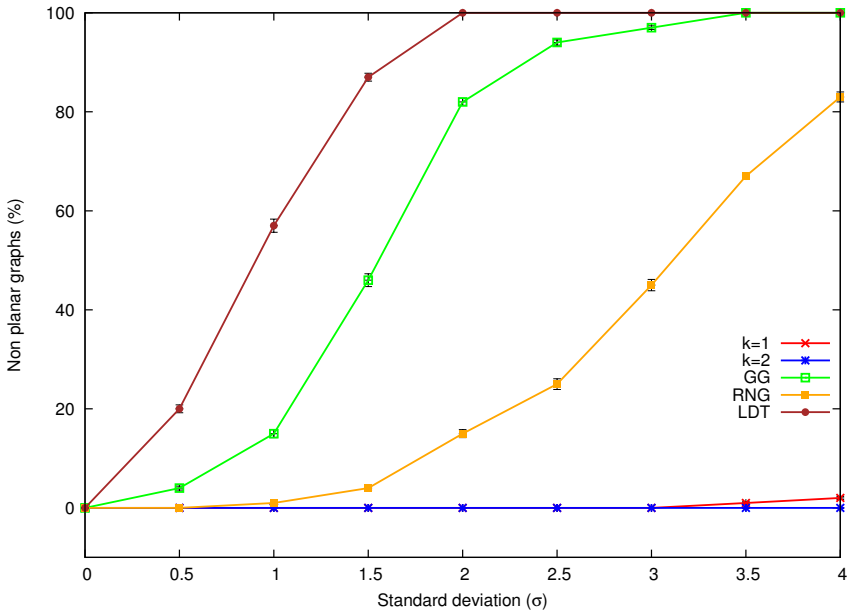


Figure 9.5: Effect of variance on planarity

σ, we could not find any direct impact of γ on planarity.

9.3.3 Effect of localization errors

To study the effect of location inaccuracy on the performance of the planarization algorithm, location errors are added to the true node positions. We used the Gaussian error model described in [154] for our analysis. There the mean value is set to 0 and standard deviation σ_{err} is varied from 0 to 20% of the *radio range*. In our experiments, we vary σ_{err} from 0% to 150% of the radio range, to find out the effect of extreme σ_{err} values. The radio range used in the LNS model is calculated using the equation 9.2, assuming $\sigma = 0$.

$$radio\ range = 10^{(P_t - PL(d_0) - RxS)/(10 * gamma))} \tag{9.2}$$

Figure 9.6 shows the results of the experiments at a confidence level of 95% on the field size 500×500 square units with $\rho \approx 3$ in the LNS networks. *LDT*, *GG* and *RNG* produce nonplanar graphs in most of the simulations when there is a small location error; but our algorithm produces planar graphs in all test cases for $\sigma_{err} < 100\%$ of the radio range. In addition to nonplanarity, location errors also cause disconnection in some networks when *LDT*, *GG* or *RNG* is used, whereas our algorithm does not disconnect any connected networks in our experiments. When the location error increases, especially when $\sigma_{err} \geq 100\%$ of the radio

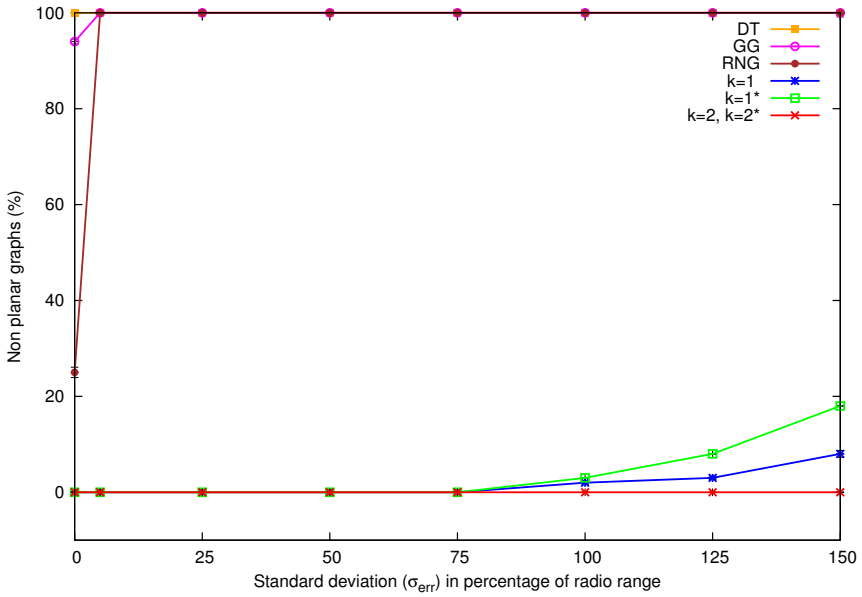


Figure 9.6: Effect of localization errors

ranges, our algorithm with $k = 1$ produces a few nonplanar graphs. The $k = 1*$ plot shows the results of our algorithm without the connectivity-threshold step, in the clustering stage. It shows that clustering with the connectivity-threshold step reduces nonplanarity.

As soon as k is increased to 2, non-planarity disappears, even at extreme location inaccuracies. Hence, in networks with extremely high node localization errors, we propose using larger cluster depths. In those networks with reasonable location accuracy, i.e. $\sigma_{err} < 100\%$ of the radio range, one hop clustering is sufficient to create planar overlay graphs.

The location fault tolerance of the planarization algorithm is due to the fact that edge removal occurs at the VCH level rather than the node level. Moreover, the position of the VCH is robust against individual location errors, especially when the cluster depth increases as the errors are averaged out. Hence the virtual link removal is less vulnerable to location errors than the physical links.

9.4 Summary

We proposed a localized planarization algorithm that creates planar graphs in realistic wireless models and which is location fault tolerant. The planarization algorithm creates an overlay graph by topology-based clustering. Using a cross link detection and repair algorithm, the intersections in the overlay graph are

removed locally. Simulation studies show that the existing localized planarization algorithms perform badly in planarizing networks with realistic wireless models, whereas this overlay-based macroscopic planarization approach proposed planarized all networks used in the simulations.

Node location errors worsen the performance of the existing localized planarization algorithms. With small location errors, $\sigma_{err} = 5\%$ of the radio range, these algorithms failed to produce any planar graphs in the simulated networks. Location errors cause disconnections too in some of the simulated networks. The planarization algorithm we proposed does not disconnect any network. Moreover, it is location fault tolerant, which is achieved by increasing the cluster depth. In networks with reasonable location accuracy, one hop clustering algorithm is sufficient to create planar overlay graphs. The additional overhead of our algorithm due to the clustering step is significantly low in this case. By increasing hop count to 2, the algorithm could achieve location fault tolerance in networks with extreme location inaccuracies as well.

CHAPTER 10

Conclusion

This dissertation described three key features of a self-organizing ad-hoc mobile robotic network, namely self-configuration, self-healing and self-optimization, and the algorithms designed for realizing such networks. In this chapter, we summarize our work and discuss some open questions and possible directions for future research.

10.1 Summary

Self-configuration capability enables the robots to deploy themselves and configure networks automatically fulfilling application requirements and self-healing capability enables them to automatically reconfigure the network in cases of failures. In AMRoNets, self-configuration is achieved through self-deployment which enables the nodes to spread out in an area maximizing coverage keeping the network connected using simple local rules. Whenever coverage or connectivity is lost, the self-healing actions are performed.

Coverage requirements vary with applications. We considered two types of coverage maximization problems in our dissertation: sensing range-based and communication range based area coverage. For sensing range-based area coverage problems we introduced a new swarm-based algorithm based on the local rules, namely separation, cohesion and alignment, used in modeling of fish schooling. Empirical analysis using simulations as well as real robot experiments conducted in various test scenarios shows that this swarm-based algorithm outperforms most prominent state-of-the-art algorithms by achieving better and faster coverage. The

algorithm could also be used for communication range-based area coverage problems, but it is not optimal in such cases.

In the communication range-based area coverage applications, AMRoNets act as a temporary infrastructure to facilitate communication between agents. Considering the slower speed of AMRoNet nodes compared to the agents and the infeasibility of a proactive pre-deployment prior to agent exploration, we proposed a simple greedy agent assisted strategy for deploying routers effectively into the area. Empirical analysis shows that the number of routers deployed by the agent-assisted router deployment algorithm is quite close to the optimal number needed in a bounded region.

Self-optimization capability enables the network to adjust regularly in varying loads and route efficiently in large-scale networks, especially when topology changes frequently. In our work we concentrate on the routing aspect of the self-optimizing networks. On analyzing the existing routing protocols, geographic routing was found to be a very interesting solution for AMRoNet routing because of its simplicity, scalability and low routing overhead.

The basic geographic routing uses a greedy forwarding step and a planar graph based FACE routing step, whenever packets cannot be forwarded according to the greedy step. FACE routing guarantees message delivery if it is applied on a planar embedding of the communication network. On looking at the various existing localized planarization algorithms, problems such as link removal even from planar graphs that could lead to disconnections have been identified. This is because of the implicit nature of the planarization algorithm. Hence we proposed a new explicit planarization algorithm called Localized Link Removal and Addition based Planarization (LLRAP) algorithm , that detects intersections locally and planarizes networks by removing them. It provably planarizes graphs more generic than UDG without causing disconnection, if the graph satisfies redundancy and coexistence properties.

Looking at the practical aspects of the geographic routing in real AMRoNet scenarios, two main challenges were identified: a) irregular radio range and b) imprecise node location information. On analyzing their impact on geographic routing, severe performance degradation has been identified. FACE routing failures attribute to most of the geographic routing failures and leads to persistent protocol failures. FACE routing failures arise from the incorrect planarization process. Localized planarization algorithms that works in real wireless networks and that are location fault tolerant are very significant for geographic routing. None of

the localized planarization algorithms including LLRAP has these characteristics. Hence we proposed a new localized planarization algorithm based on a topological cluster-based overlay graph construction.

The planarization algorithm creates an overlay graph by topology-based clustering. As overlay graphs may contain intersections, to make them planar, explicit planarization algorithms are selected as they are more suitable than implicit algorithms in such non UDGs. A tailored version of the LLRAP algorithm called Cross Link Detection and Repair has been applied to remove intersections in the overlay graph locally. Empirical analysis shows that this overlay-based macroscopic planarization approach is location fault tolerant and produces planar graphs.

10.2 Future Directions

The swarm-based algorithm works well in simulations and Teleworkbench where the position information is available readily. A real world test of this approach would be highly interesting, especially relying only on the dead reckoning based position information and dealing with localization errors. Testing the agent assisted router deployment algorithm relying only on link quality estimate is yet to be done.

The sensing range-based area coverage is a well-studied problem, but only few works concentrate on the communication range-based area coverage problem. In the communication range-based area coverage applications, when a proactive predeployment prior to agent exploration is feasible, the self-deployment of the nodes is an interesting question. We could extend the agent-assisted router deployment algorithm to develop a self-spreading algorithm for such scenarios, but it may not be the best solution. A localized self-deploying algorithm that has good coverage metrics is yet to be done.

Considering the self-optimized routing, the specification of a complete routing protocol needs to be done. We could use two different routing modes, an inter-cluster mode and an intra-cluster mode. In the intra-cluster routing mode, nodes could use greedy forwarding to send packets to their destinations. For inter-cluster routing, nodes could send packets to those VCHs in their neighborhood that minimizes the forwarding metric. At a void VCH, planar graph routing along the virtual faces of the overlay graph could be employed.

Besides the routing protocol, we plan to make theoretical analysis about the planarity of overlay graphs created from the topological clustering in restricted

network models such as UDG or d-QUDG with $d \geq \frac{1}{\sqrt{2}}$. We also plan to investigate the potential improvements that could be achieved with our algorithm in reducing the overhead of current location services by keeping virtual cluster heads instead of real nodes. This needs to be validated with more quantitative results.

APPENDIX A

Player/Stage

A.1 Player Configuration file

```
# File:bebot.cfg
# Desc: Player sample configuration file for controlling Stage devices
# Load the Stage plugin simulation driver
driver
(
  name "stage"
  provides [ "simulation:0" ]
  plugin "stageplugin"

  # load the named file into the simulator
  worldfile "bebot.world"
)

# Create a Stage driver and attach position2d and ranger interfaces
# to the model "bebot"
driver
(
  name "stage"
  provides [ "6665:position2d:0" "6665:ranger:0" "6665:wifi:0"]
  model "bebot-1"
)
.....
```

```
driver
(
  name "stage"
  provides [ "6669:position2d:0" "6669:ranger:0" "6669:wifi:0"]
  model "bebot-55"
)
```

A.2 World file

```
# File: bebot.world

include "bebot.inc"
include "map.inc"
include "nbebot.inc"

# time to pause (in GUI mode) or quit (in headless mode (-g)) the simulation
# quit_time 3600    # 1 hour of simulated time
# interval_sim 100  # simulation timestep in milliseconds
# interval_real 250 # real-time interval between simulation updates in milliseconds

paused 0
resolution 0.005

# configure the GUI window
window
(
  size [ 1300.000 700.000 ] # in pixels
  scale 50                  # pixels per meter
  show_data 1               # 1=on 0=off
# center [ -0.040  -0.274 ]
# rotate [ 0  0 ]
)

# load an environment bitmap
floorplan
(
```

```
  name "hospital_section"
  boundary 1
  size [ 24 12 0.3 ]
  pose [0 0 0 0]
  bitmap "bitmaps/cage.png"
)

# File: bebot.inc
# Desc: The bebot with IR array

# IR sensors
define ir sensor
(
  # define the size of each transducer [xsize ysize zsize] in meters
  size [0.001 0.002 0.001 ]
  # define the range bounds [min max]
  range [0 0.5]
  # define the angular field of view in degrees
  fov 30
  # define the number of samples spread over the fov
  samples 1
  # define the color that ranges are drawn in the gui
  color_rgba [ 0 1 0 0.2 ]

)

define bebot_ir ranger
(

 # 12 transducers spread about the robot as follows
 # define the pose of each sensor [xpos ypos zpos heading]
 ir( pose [  0.0455  0.014 0   15 ] )
 ir( pose [  0.0425  0.041 0   45 ] )
 ir( pose [  0.0195  0.044 0   75 ] )
 ir( pose [ -0.0195  0.044 0  105 ] )
 ir( pose [ -0.0425  0.041 0  135 ] )
 ir( pose [ -0.0455  0.014 0  165 ] )
```

```
 ir( pose [ -0.0455 -0.014 0 -165 ] )
 ir( pose [ -0.0425 -0.041 0 -135 ] )
 ir( pose [ -0.0195 -0.044 0 -105 ] )
 ir( pose [  0.0195 -0.044 0  -75 ] )
 ir( pose [  0.0425 -0.041 0  -45 ] )
 ir( pose [  0.0455 -0.014 0  -15 ] )
)

define bebot_base position
(
  color "red"     # Default color.
  drive "diff"     # Differential steering model.
  gui_nose 1     # Draw a nose on the robot so we can see which way it points
  obstacle_return 1 # Can hit things.
  ranger_return 0.5 # reflects sonar beams
  blob_return 1     # Seen by blobfinders
  fiducial_return 1 # Seen as "1" fiducial finders

  localization "gps"
  localization_origin [0 0 0 0] # Start odometry at (0, 0, 0).

  # alternative odometric localization with simple error model
  # localization "odom"
  # Change to "gps" to have impossibly perfect, global odometry
  # odom_error [ 0.05 0.05 0.1 ]
  # Odometry error or slip in X, Y and Theta (Uniform random distribution)
)



define bebot bebot_base
(
  # actual size
  size [0.091 0.088 0.09]

  # center of region
```

```
  origin [0 0 0 0]

  # draw a nose on the robot so we can see which way it points
  gui_nose 1

  # estimated mass in KG
  mass 1.0

  # differential steering model
  drive "diff"

  block
  (
    points 8
    point[7] [ -0.0455  0.0390 ]
    point[6] [ -0.0450  0.0440 ]
    point[5] [  0.0450  0.0440 ]
    point[4] [  0.0455  0.0390 ]
    point[3] [  0.0455 -0.0390 ]
    point[2] [  0.0450 -0.0440 ]
    point[1] [ -0.0450 -0.0440 ]
    point[0] [ -0.0455 -0.0390 ]
    z [0.037 0.090]
    color "gray50"
  )

  bebot_ir( pose [0 0 -0.03 0.8] )
)

# File: nbebot.inc

bebot
(
  # can refer to the robot by this name
  name "bebot-1"
  pose [ 1.84812 1.99941 0 0 ]
  # report error-free position in world coordinates
```

```
  localization "gps"
  localization_origin [ 0 0 0 0 ]
  wifi(
    ip "192.168.0.1"
    mac "09:56:45:ae:ae:01"
    essid "bebot netwok"
    model "simple"
    range 1
    )
  )

.....

bebot
(
  # can refer to the robot by this name
  name "bebot-55"
  pose [ 6.955 1.99623 0 0 ]
  # report error-free position in world coordinates
  localization "gps"
  localization_origin [ 0 0 0 0 ]
  wifi(
    ip "192.168.0.55"
    mac "09:56:45:ae:ae:55"
    essid "bebot netwok"
    model "simple"
    range 1
    )
  )


# File: map.inc -
# Desc: Useful setup for a floorplan bitmap
# Authors: Richard Vaughan

define floorplan model
(
```

```
  # sombre, sensible, artistic
  color "gray30"

  # most maps will need a bounding box
  #boundary 1

  gui_nose 0
  gui_grid 0
  gui_move 0
  gui_outline 0
  gripper_return 0
  fiducial_return 0
  ranger_return 0.5
)

define zone model
(
  color "orange"
  size [ 2 2 0.02 ]

  gui_nose 0
  gui_grid 0
  gui_move 1
  gui_outline 0

  # insensible to collision and range sensors
  obstacle_return 0
  ranger_return -1  # transparent to range sensors
)
```

Own publications

[1] Peter Janacik, Emi Mathews, and Dalimir Orfanus. Self-organizing data collection in wireless sensor networks. *International Conference on Advanced Information Networking and Applications Workshops*, pages 662–667, 2010.

[2] Yara Khaluf, Emi Mathews, and Franz Josef Rammig. Self-organized cooperation in swarm robotics. In *14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW)*, pages 217–226, California, USA, March 2011. IEEE, IEEE Computer Society.

[3] Yara Khaluf, Emi Mathews, and Franz Josef Rammig. Swarm robotic time synchronization for object tracking. In Teresa Higuera, Uwe Brinkschulte, and Achim Rettberg, editors, *Self-Organization in Embedded Real-Time Systems*. Springer, 2012, to appear.

[4] Emi Mathews. Maintaining connectivity of autonomous agents using mobile ad-hoc robotic network. In Kai Bollue, Dominique Gückel, Ulrich Loup, Jacob Spönemann, and Melanie Winkler, editors, *Proceedings of the Joint Workshop of the German Research Training Groups in Computer Science, Algorithmic synthesis of reactive and discrete-continuous systems (AlgoSyn 2010)*, page 187, Dagstuhl, Germany, May-June 2010.

[5] Emi Mathews. Planarization of geographic cluster-based overlay graphs in realistic wireless networks. In *9th International Conference on Information Technology : New Generations (ITNG)*, Las Vegas, USA, April 2012. IEEE Computer Society.

[6] Emi Mathews and Hannes Frey. Topological cluster based geographic routing in multihop ad hoc networks. In *The Fourth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, UBICOMM*, October 2010.

[7] Emi Mathews and Hannes Frey. A localized planarization algorithm for realistic wireless networks. In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–9, Lucca, Italy, June 2011. IEEE Computer Society.

[8] Emi Mathews and Hannes Frey. A localized link removal and addition based planarization algorithm. In *13th International Conference on Distributed Computing and Networking (ICDCN)*, Lecture Notes in Computer Science, Hong Kong, China, 2012. Springer.

[9] Emi Mathews, Tobias Graf, and K.S.S.B Kulathunga. A bio-inspired coverage and connectivity maintenance algorithm. In *Sixth International ICST Conference on Bio-Inspired Models of Network, Information, and Computing Systems*, York, UK, December 2011. Lecture Notes of ICST.

[10] Emi Mathews, Tobias Graf, and K.S.S.B Kulathunga. Biologically inspired swarm robotic network ensuring coverage and connectivity. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC 2012)*, October, to appear 2012.

[11] Emi Mathews and Ciby Mathew. Connectivity of autonomous agents using ad-hoc mobile router networks. In *Third International Conference on Networks and Communications*, LNICST, Bangalore, India, January 2012. Springer.

[12] Emi Mathews and Ciby Mathew. Deployment of mobile routers ensuring coverage and connectivity. *International Journal of Computer Networks and Communications*, 4, January 2012.

Bibliography

[13] Ameer Ahmed Abbasi and Mohamed Younis. A survey on clustering algorithms for wireless sensor networks. *Computer Communications*, 30(14-15):2826–2841, October 2007.

[14] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393 – 422, 2002.

[15] Ronald C. Arkin and Khaled S. Ali. Integration of reactive and telerobotic control in multi-agent robotic systems. In *Third International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, pages 473–478. MIT Press, 1994.

[16] Xiaole Bai, Santosh Kumar, Dong Xuan, Ziqiu Yun, and Ten H. Lai. Deploying wireless sensors to achieve both coverage and connectivity. In *Seventh ACM international symposium on Mobile ad hoc networking and computing (MobiHoc 2006)*, pages 131–142, New York, NY, USA, 2006. ACM.

[17] Tucker Balch and Maria Hybinette. Behavior-based coordination of large-scale robot formations. In *Fourth International Conference on MultiAgent Systems, ICMAS-2000*, pages 363–364, Washington, DC, USA, 2000. IEEE Computer Society.

[18] M. Ballerini, N. Cabibbo, R. Candelier, A. Cavagna, E. Cisbani, I. Giardina, V. Lecomte, A. Orlandi, G. Parisi, A. Procaccini, M. Viale, and V. Zdravkovic. Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study. *Proceedings of the National Academy of Sciences*, 105(4):1232–1237, 2008.

[19] Lali Barriére, Pierre Fraigniaud, and Lata Narayanan. Robust position-based routing in wireless ad hoc networks with unstable transmission ranges. In *Fifth international workshop on Discrete algorithms and methods for mobile computing and communications (DIALM 2001)*, pages 19–27, New York, NY, USA, 2001. ACM.

[20] Lali Barriére, Pierre Fraigniaud, Lata Narayanan, and Jaroslav Opatrny. Robust position-based routing in wireless ad hoc networks with irregular transmission ranges. *Wireless Communications and Mobile Computing*, 3(2):141–153, 2003.

[21] M.A. Batalin and G.S. Sukhatme. The design and analysis of an efficient local algorithm for coverage and exploration based on sensor network deployment. *IEEE Transactions on Robotics*, 23(4):661 –675, August 2007.

[22] Maxim A. Batalin and Gaurav S. Sukhatme. Spreading out: A local approach to multi-robot coverage. In *Sixth International Symposium on Distributed Autonomous Robotic Systems*, pages 373–382, 2002.

[23] Geoffrey Biggs and Bruce Macdonald. A survey of robot programming systems. In *Australasian Conference on Robotics and Automation, CSIRO*, page 27, 2003.

[24] Ljubica Blažević, Silvia Giordano, and Jean-Yves Le Boudec. Self organized terminode routing. *Cluster Computing*, 5(2):205–218, 2002.

[25] A. Bogdanov, E. Maneva, and S. Riesenfeld. Power-aware base station positioning for sensor networks. In *Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004)*, volume 1, March 2004.

[26] Prosenjit Bose, Pat Morin, Ivan Stojmenović, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Third international workshop on Discrete algorithms and methods for mobile computing*

and communications (DIALM 1999), pages 48–55, New York, NY, USA, 1999. ACM.

[27] Jonathan L. Bredin, Erik D. Demaine, MohammadTaghi Hajiaghayi, and Daniela Rus. Deploying sensor networks with guaranteed capacity and fault tolerance. In *Sixth ACM international symposium on Mobile ad hoc networking and computing (MobiHoc 2005)*, pages 309–319, New York, NY, USA, 2005. ACM.

[28] Wolfram Burgard, Mark Moors, Cyrill Stachniss, and Frank Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21:376–386, 2005.

[29] Jiannong Cao, Lifan Zhang, Guojun Wang, and Hui Cheng. SSR: Segment-by-segment routing in large-scale mobile ad hoc networks. In *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*, volume 0, pages 216–225, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[30] Dazhi Chen, Jing Deng, and P.K. Varshney. On the forwarding area of contention-based geographic forwarding for ad hoc and sensor networks. In *Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2005)*, pages 130–141, September 2005.

[31] Dazhi Chen, Jing Deng, and P.K. Varshney. A state-free data delivery protocol for multihop wireless sensor networks. In *IEEE Wireless Communications and Networking Conference (WCNC 2005)*, volume 3, pages 1818–1823, March 2005.

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

[33] Dazhi Chen and Pramod K. Varshney. On-demand geographic forwarding for data delivery in wireless sensor networks. *Comput. Commun.*, 30:2954–2967, October 2007.

[34] Dazhi Chen and Pramod K. Varshney. *Geographic Routing in Wireless Ad Hoc Networks*, pages 1–38. Springer London, 2009.

[35] Geng Chen, Fabian Garcia Nocetti, Julio Solano Gonzalez, and Ivan Stojmenović. Connectivity-based k-hop clustering in wireless networks. In *Thirty fifth Annual Hawaii International Conference on System Sciences (HICSS 2002)*, volume 7, pages 205–217, Washington, DC, USA, 2002.

[36] Shigang Chen, Guangbin Fan, and Jun Hong Cui. Avoid void in geographic routing for data aggregation in sensor networks. *Int. J. Ad Hoc Ubiquitous Comput.*, 1(4):169–178, July 2006.

[37] Ching-Chuan Chiang, Hsiao-Kuang Wu, Winston Liu, and Mario Gerla. Routing in clustered multihop mobile wireless networks with fading channel. In *IEEE Singapore International Conference on Networks (SICON 1997)*, pages 197–211, April 1997.

[38] Howie Choset. Coverage for robotics - a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):113–126, May 2001.

[39] Hedrick C.L. Routing information protocol. RFC 1058 (Historic), June 1988.

[40] Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1-3):165–177, 1990.

[41] Nikolaus Correll and Alcherio Martinoli. Robust distributed coverage using a swarm of miniature robots. In *IEEE International Conference on Robotics and Automation (ICRA 2007)*, pages 379–384, 2007.

[42] D. R. Cox and D. V Hinkley. *Theoretical statistics*. Chapman and Hall London, 1974.

[43] Koustuv Dasgupta, Meghna Kukreja, and Konstantinos Kalpakis. Topology-aware placement and role assignment for energy-efficient information gathering in sensor networks. In *Eighth IEEE International Symposium on Computers and Communications (ISCC 2003)*, pages 341–354, Washington, DC, USA, 2003. IEEE Computer Society.

[44] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, second edition, 2000.

[45] Göksel Dedeoglu and Gaurav Sukhatme. Landmark-based matching algorithm for cooperative mapping by autonomous robots. In *Distributed Autonomous Robotics Systems*, pages 251–260. Springer-Verlag, 2000.

[46] S.S. Dhillon and K. Chakrabarty. Sensor placement for effective coverage and surveillance in distributed sensor networks. In *Wireless Communications and Networking (WCNC 2003)*, volume 3, pages 1609 –1614, March 2003.

[47] Marco Dorigo, Elio Tuci, Vito Trianni, Roderich Groß, Shervin Nouyan, Christos Ampatzis, Thomas Halva Labella, Rehan O'Grady, Michael Bonani, and Francesco Mondada. SWARM-BOT: Design and implementation of colonies of self-assembling robots. In G. Y. Yen and D. B. Fogel, editors, *Computational Intelligence: Principles and Practice*, chapter 6, pages 103–135. IEEE Computational Intelligence Society, New York, 2006.

[48] Miroslaw Dynia, Jaroslaw Kutylowski, Friedhelm Meyer auf der Heide, and Jonas Schrieb. Local strategies for maintaining a chain of relay stations between an explorer and a base station (spaa 2007). In *Nineteenth annual ACM symposium on Parallel algorithms and architectures*, pages 260–269, New York, NY, USA, 2007. ACM.

[49] A. Efrat, S. Har-Peled, and J.S.B. Mitchell. Approximation algorithms for two optimal location problems in sensor networks. In *SecondInternational Conference on Broadband Networks (BroadNets 2005)*, volume 1, pages 714–723, October 2005.

[50] Qing Fang, Jie Gao, and Leonidas J. Guibas. Locating and bypassing holes in sensor networks. *Mobile Networks and Applications*, 11:187–200, April 2006.

[51] Qing Fang, Jie Gao, and L.J. Guibas. Locating and bypassing routing holes in sensor networks. In *Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004)*, volume 4, pages 2458–2468, March 2004.

[52] Qing Fang, Jie Gao, L.J. Guibas, V. de Silva, and Li Zhang. Glider: gradient landmark-based distributed routing for sensor networks. In *IEEE International Conference on Computer Communications (INFOCOM 2005)*, volume 1, pages 339–350, March 2005.

[53] D. Ferrara, L. Galluccio, A. Leonardi, G. Morabito, and S. Palazzo. MACRO: an integrated MAC/routing protocol for geographic forwarding in wireless sensor networks. In *Twenty fourth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005)*, volume 3, pages 1770–1781, March 2005.

[54] Gregory G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical report, University of Southern California, 1987.

[55] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *National Conference on Artificial Intelligence*, 1999.

[56] Jakob Fredslund and Maja J. Mataric. Robot formations using only local sensing and control. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA 2001)*, pages 308–313, 2001.

[57] Hannes Frey. Geographical cluster based multihop ad hoc network routing with guaranteed delivery. *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*, pages 509–519, 2005.

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

[59] Hannes Frey and Stefan Rührup. Paving the way towards reactive planar spanner construction in wireless networks. In *Kommunikation in Verteilten Systemen (KiVS)*, Informatik aktuell, pages 17–28. Springer Berlin Heidelberg, 2009.

[60] Hannes Frey and Ivan Stojmenović. On delivery guarantees of face and combined greedy-face routing in ad hoc and sensor networks. In *Twelth annual international conference on Mobile computing and networking (MobiCom 2006)*, pages 390–401, New York, NY, USA, 2006. ACM.

[61] Holger Freyther, Richard Purdie, and Chris Larson. Bitbake build tool. Accessed on July 1, 2012.

[62] R. Friedman and G. Kliot. Location services in wireless ad hoc and hybrid networks: A survey. Technical report, Israel Institute of Technology, Haifa, Israel, 2006.

[63] Stefan Funke and Nikola Milosavljević. Guaranteed-delivery geographic routing under uncertain node locations. In *IEEE International Conference on Computer Communications (INFOCOM 2007)*, pages 1244 –1252, May 2007.

[64] H. Fussler, J. Widmer, M. Mauve, and H. Hartenstein. A novel forwarding paradigm for position-based routing (with implicit addressing). In *IEEE Eighteenth Annual Workshop on Computer Communications (CCW 2003)*, pages 194–200, October 2003.

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

[66] Y. Gabriely and E. Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. In *IEEE International Conference on Robotics and Automation (ICRA 2001)*, volume 2, pages 1927–1933, 2001.

[67] Douglas W. Gage. Command control for many-robot systems. *Nineteenth Annual AUVS Technical Symposium (AUVS 1992), Reprinted in Unmanned Systems Magazine*, 10(4):28–34, June 1992.

[68] Deepak Ganesan, Răzvan Cristescu, and Baltasar Beferull-Lozano. Power-efficient sensor placement and transmission structure for data gathering under distortion constraints. *ACM Transactions on Sensor Networks*, 2(2):155–181, May 2006.

[69] Jie Gao, Leonidas J. Guibas, John Hershberger, Li Zhang, and An Zhu. Geometric spanner for routing in mobile networks. In *Second ACM international symposium on Mobile ad hoc networking & computing (MobiHoc 2001)*, pages 45–55, New York, NY, USA, 2001. ACM.

[70] J. J. Garcia-Luna-Aceves and Marcelo Spohn. Source-tree routing in wireless networks. In *Seventh Annual International Conference on Network Protocols (ICNP 1999)*, pages 273–283, Washington, DC, USA, 1999. IEEE Computer Society.

[71] V. Genovese, P. Dario, R. Magni, and L. Odetti. Self organizing behavior and swarm intelligence in a pack of mobile miniature robots in search of

pollutants. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 1992)*, volume 3, pages 1575–1582, July 1992.

[72] Brian Gerkey, Richard T. Vaughan, and Andrew Howard. The player project. Accessed on July 1, 2012.

[73] Brian Gerkey, Richard T. Vaughan, and Andrew Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Eleventh International Conference on Advanced Robotics (ICAR 2003)*, pages 317–323, New York, NY, USA, June 2003.

[74] Amitabha Ghosh and Sajal K. Das. Coverage and connectivity issues in wireless sensor networks: A survey. *Pervasive and Mobile Computing*, 4(3):303 – 334, 2008.

[75] Young-Jin Kim Ramesh Govindan, Brad Karp, and Scott Shenker. Lazy cross-link removal for geographic routing. In *Fourth international conference on Embedded networked sensor systems (SenSys 2006)*, pages 112–124, New York, NY, USA, 2006. ACM.

[76] Z.J. Haas. A new routing protocol for the reconfigurable wireless networks. In *IEEE 6th International Conference on Universal Personal Communications (ICUPC 1997)*, volume 2, pages 562–566, October 1997.

[77] Sabine Hauert, Laurent Winkler, Jean-Christophe Zufferey, and Dario Floreano. Ant-based swarming with positionless micro air vehicles for communication relay. *Swarm Intelligence*, 2(2-4):167–188, 2008.

[78] N. Hazon and G.A. Kaminka. Redundancy, efficiency and robustness in multi-robot coverage. In *IEEE International Conference on Robotics and Automation (ICRA 2005)*, pages 735 – 741, April 2005.

[79] N. Hazon, F. Mieli, and G.A. Kaminka. Towards robust on-line multi-robot coverage. In *IEEE International Conference on Robotics and Automation (ICRA 2006)*, pages 1710 –1715, May 2006.

[80] Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, and Tarek Abdelzaher. Range-free localization schemes for large scale sensor networks. In *Ninth Annual International Conference on Mobile Computing and Networking (MobiCom 2003)*, pages 81–95, New York, NY, USA, 2003. ACM.

[81] Marc Heissenbüttel, Torsten Braun, Thomas Bernoulli, and Markus Wälchli. Blr: beacon-less routing algorithm for mobile ad hoc networks. *Computer Communications*, 27(11):1076–1086, 2004. Applications and Services in Wireless Networks.

[82] N. Heo and P.K. Varshney. A distributed self spreading algorithm for mobile wireless sensor networks. In *IEEE Wireless Communications and Networking (WCNC 2003)*, volume 3, pages 1597 –1602, march 2003.

[83] Stefan Herbrechtsmeier, Ulf Witkowski, and Ulrich Rückert. Bebot: A modular mobile miniature robot platform supporting hardware reconfiguration and multi-standard communication. In *Progress in Robotics*, volume 44 of *Communications in Computer and Information Science*, pages 346–356. Springer Berlin Heidelberg, 2009.

[84] Jeffrey Hightower and Gaetano Borriella. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, August 2001.

[85] Bernhard Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning System: Theory and Practice*. Springer-Verlag, 1997.

[86] Ting-Chao Hou and Victor Li. Transmission range control in multihop packet radio networks. *IEEE Transactions on Communications*, 34(1):38–44, January 1986.

[87] Y.T. Hou, Yi Shi, H.D. Sherali, and S.F. Midkiff. On energy provisioning and relay node placement for wireless sensor networks. *IEEE Transactions on Wireless Communications*, 4(5):2579–2590, September 2005.

[88] Andrew Howard, Maja J. Matarić, and Gaurav S. Sukhatme. An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robots Special Issue on Intelligent Embedded Systems*, 13(2):113–126, 2002.

[89] Andrew Howard, Maja J Mataric, and Gaurav S Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Sixth International Symposium on Distributed Autonomous Robotics Systems (DARS 2002)*, pages 299–308, June 2002.

[90] M. J. Howard Andrew, Mataric. Cover me! a self-deployment algorithm for mobile sensor networks. In *IEEE International Conference on Robotics and Automation (ICRA 2002)*, May 2002.

[91] T. Hsiang, E. Arkin, M. Bender, S. Fekete, and J. Mitchell. Algorithms for rapidly dispersing robot swarms in unknown environments. In *Proceedings of the Algorithmic Foundations of Robotics*, pages 77–94, Berlin, Germany, 2002. Springer.

[92] Mika Ishizuka and Masaki Aida. Performance study of node placement in sensor networks. In *Twenty fourth International Conference on Distributed Computing Systems Workshops (ICDCSW 2004)*, volume 7, pages 598–603, Washington, DC, USA, 2004. IEEE Computer Society.

[93] A. Iwata, Ching-Chuan Chiang, Guangyu Pei, M. Gerla, and Tsu-Wei Chen. Scalable routing strategies for ad hoc wireless networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1369–1379, August 1999.

[94] Rajagopal Iyengar, Koushik Kar, and Suman Banerjee. Low-coordination topologies for redundancy in sensor networks. In *Sixth ACM international symposium on Mobile ad hoc networking and computing (MobiHoc 2005)*, pages 332–342, New York, NY, USA, 2005. ACM.

[95] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. In *IEEE Multi Topic Conference, Technology for the 21st Century Proceedings*, pages 62–68, 2001.

[96] M. Joa-Ng and I-Tai Lu. A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1415–1425, August 1999.

[97] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

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

[99] Brad Karp. Challenges in geographic routing: Sparse networks, obstacles, and traffic provisioning. In *DIMACS Workshop on Pervasive Networking*, May 2001.

[100] Brad Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Sixth annual international conference on Mobile computing and networking (MobiCom 2000)*, pages 243–254, New York, USA, 2000. ACM.

[101] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, April 1986.

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

[103] Young-Jin Kim, Ramesh Govindan, Brad Karp, and Scott Shenker. Geographic routing made practical. In *Second conference on Symposium on Networked Systems Design & Implementation (NSDI 2005)*, pages 217–230, Berkeley, CA, USA, 2005. USENIX Association.

[104] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. *Wireless Networks*, 6(4):307–321, July 2000.

[105] Sven Koenig, Boleslaw Szymanski, and Yaxin Liu. Efficient and inefficient ant coverage methods. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):41–76, May 2001.

[106] Evangelos Kranakis, Harvinder Singh, and Jorge Urrutia. Compass routing on geometric networks. In *Eleventh Canadian Conference on Computational Geometry (CCCG 1999)*, pages 51–54, August 1999.

[107] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *Fourth ACM International Symposium on Mobile ad hoc networking & computing (MobiHoc 2003)*, pages 267–278, New York, NY, USA, 2003. ACM.

[108] Fabian Kuhn, Rogert Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: of theory and practice. In *Twenty-second annual symposium on Principles of distributed computing (PODC 2003)*, pages 63–72, New York, NY, USA, 2003. ACM.

[109] Fabian Kuhn, Rogert Wattenhofer, and Aaron Zollinger. Asymptotically optimal geometric mobile ad-hoc routing. In *Sixth International Workshop on Discrete algorithms and methods for mobile computing and communications (DIALM 2002)*, pages 24–33, New York, NY, USA, 2002. ACM.

[110] Fabian Kuhn and Aaron Zollinger. Ad-hoc networks beyond unit disk graphs. In *Joint workshop on Foundations of mobile computing (DIALM-POMC 2003)*, pages 69–78, New York, NY, USA, 2003. ACM.

[111] Johnson Kuruvila, Amiya Nayak, and Ivan Stojmenovic. Greedy localized routing for maximizing probability of delivery in wireless ad hoc networks with a realistic physical layer. *Journal of Parallel and Distributed Computing*, 66(4):499–506, April 2006.

[112] Jarosaw Kutyowski and Friedhelm Meyer auf der Heide. Optimal strategies for maintaining a chain of relays between an explorer and a base camp. *Theoretical Computer Science*, 410:3391–3405, August 2009.

[113] Chris Larson, Michael Lauer, and Holger Schurig. Openembedded – the build framework for embedded linux. Accessed on July 1, 2012.

[114] D. Laselva, Xiongwen Zhao, J. Meinila, T. Jamsa, J.-P. Nuutinen, P. Kyosti, and L. Hentila. Empirical models and parameters for rural and indoor wideband radio channels at 2.45 and 5.25 GHZ. *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 1:654–658, 2005.

[115] Seungjoon Lee, Bobby Bhattacharjee, and Suman Banerjee. Efficient geographic routing in multihop wireless networks. In *Sixth ACM international symposium on Mobile ad hoc networking and computing (MobiHoc 2005)*, pages 230–241, New York, NY, USA, 2005. ACM.

[116] Ben Leong, Barbara Liskov, and Robert Morris. Geographic routing without planarization. In *Symposium on Network Systems Design and Implementation (NSDI 2006)*, pages 339–352, 2006.

[117] Ben Leong, Sayan Mitra, and Barbara Liskov. Path vector face routing: Geographic routing with local face information. In *Thirteenth IEEE International Conference on Network Protocols (ICNP 2005)*, pages 147–158, Washington, DC, USA, 2005. IEEE Computer Society.

[118] Johannes Lessmann, Tales Heimfarth, and Peter Janacik. Shox: An easy to use simulation platform for wireless networks. In *Tenth International Conference on Computer Modeling and Simulation (UKSIM)*, pages 410–415, 2008.

[119] N. Li, J.C. Hou, and L. Sha. Design and analysis of an mst-based topology control algorithm. In *Twenty second Annual Joint Conference of the IEEE Computer and Communications (INFOCOM 2003)*, volume 3, pages 1702 – in1712, March-April 2003.

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

[121] Wen-Hwa Liao, Jang-Ping Sheu, and Yu-Chee Tseng. Grid: A fully location-aware routing protocol for mobile ad hoc networks. *Telecommunication Systems*, pages 37–60, 2001.

[122] Kevin M. Lillis, Sriram V. Pemmaraju, and Imran A. Pirwani. Topology control and geographic routing in realistic wireless networks. *Ad Hoc & Sensor Wireless Networks*, 6(3-4):265–297, 2008.

[123] Junlong Lin and Geng-Sheng Kuo. A novel location-fault-tolerant geographic routing scheme for wireless ad hoc networks. In *IEEE International Conference on Vehicular Technology Conference*, volume 3, pages 1092 – 1096, Melbourne, Australia, 2006. IEEE Computer Society.

[124] Errol L. Lloyd and Guoliang Xue. Relay node placement in wireless sensor networks. *IEEE Transactions Computers*, 56(1):134–138, January 2007.

[125] Maite López-Sánchez, Francesc Esteva, de Mántaras López de Mantaras, Carles Sierra, and Josep Amat. Map generation by cooperative low-cost robots in structured unknown environments. *Autonomous Robots*, 5(1):53–61, 1998.

[126] Luke Ludwig and Maria Gini. Robotic swarm dispersion using wireless intensity signals. In *International Symposium on Distributed Autonomous Robotic Systems*, 2006.

[127] Maja J. Mataric. Designing and understanding adaptive group behavior. *Adaptive Behavior*, 4:51–80, 1995.

[128] J McLurkin and J Smith. Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots. *Seventh International Symposium on Distributed Autonomous Robotic Systems (DARS 2004)*, 2004.

[129] John M. McQuillan and David C. Walden. The ARPA network design decisions. *Computer Networks*, 1:243–289, 1977.

[130] Ryan Morlok and Maria Gini. Dispersing robots in an unknown environment. In *Seventh International Symposium on Distributed Autonomous Robotic Systems (DARS 2004)*, 2004.

[131] T. Muetze, P. Stuedi, F. Kuhn, and G. Alonso. Understanding radio irregularity in wireless networks. In *Fifth Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2008)*, pages 82 –90, June 2008.

[132] C. Siva Ram Murthy and B.S. Manoj. *Ad Hoc Wireless Networks: Architectures and Protocols*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[133] Shree Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *Mobile Networks and Applications*, 1(2):183–197, October 1996.

[134] Bob O'Hara and Al Petrick. *The IEEE 802.11 Handbook: A Designer's Companion*. Standards Information Network IEEE Press, 1999.

[135] Joseph O'Rourke. *Art Gallery Theorem and Algorithms*. New York: Oxford University Press, 1987.

[136] Eliyahu Osherovich, Vladimir Yanovski, Israel A. Wagner, and Alfred M. Bruckstein. Robust and efficient covering of unknown continuous domains with simple, ant-like a(ge)nts. *The International Journal of Robotics Research*, 27(7):815–831, July 2008.

[137] Anuraag Pakanati and Maria Gini. Swarm dispersion via potential fields, leader election, and counting hops. In *Second international conference on Simulation, modeling, and programming for autonomous robots (SIMPAR 2010)*, pages 485–496. Springer-Verlag, 2010.

[138] Jianping Pan, Lin Cai, Y. Thomas Hou, Yi Shi, and Sherman X. Shen. Optimal base-station locations in two-tiered wireless sensor networks. *IEEE Transactions on Mobile Computing*, 4(5):458–473, September 2005.

[139] V.D. Park and M.S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 1997)*, volume 3, pages 1405–1413, April 1997.

[140] David W. Payton, Mike Daily, Regina Estkowski, Mike Howard, and Craig Lee. Pheromone robotic. *Autonomous Robots*, 11(3):319–324, 2001.

[141] J.L. Pearce, P.E. Rybski, S.A. Stoeter, and N. Papanikolopoulos. Dispersion behaviors for a team of multiple miniature robots. In *IEEE International Conference on Robotics and Automation (ICRA 2003)*, volume 1, pages 1158–1163, September 2003.

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

[143] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *SIGCOMM Computer Communication Review*, 24(4):234–244, October 1994.

[144] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. In *Second IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1997.

[145] Bue Petersen and Jonas Fonseca. Player/stage - player driver implementation for ersp. Technical report, Department of Computer Science, University of Copenhagen, 2006.

[146] S.J. Philip, Joy Ghosh, H.Q. Ngo, and Chunming Qiao. Routing on overlay graphs in mobile ad hoc networks. In *IEEE Global Telecommunications Conference (GLOBECOM 2006)*, pages 1–6, December 2006.

[147] Sameera Poduri and Gaurav S. Sukhatme. Constrained coverage for mobile sensor networks. In *IEEE International Conference on Robotics and Automation*, pages 165–172, New Orleans, LA, May 2004.

[148] Dario Pompili, Tommaso Melodia, and Ian F. Akyildiz. Deployment analysis in underwater acoustic wireless sensor networks. In *First ACM international workshop on Underwater networks (WUWNet 2006)*, pages 48–55, New York, NY, USA, 2006. ACM.

[149] Theodore Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 2001.

[150] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Computer Graphics*, 21:25–34, 1987.

[151] Martijn N. Rooker and Andreas Birk. Multi-robot exploration under the constraints of wireless networking. *Control Engineering Practice*, 15(4):435 – 445, 2007.

[152] Stefan Rührup. *Theory and Practice of Geographic Routing*. Bentham Science, 2009.

[153] Christian Schindelhauer, Klaus Volbert, and Martin Ziegler. Spanners, weak spanners, and power spanners for wireless networks. In *Fifteenth international conference on Algorithms and Computation (ISAAC 2004)*, pages 805–821, Berlin, Heidelberg, 2004. Springer-Verlag.

[154] Karim Seada, Ahmed Helmy, and Ramesh Govindan. On the effect of localization errors on geographic face routing in sensor networks. In *Third international symposium on Information processing in sensor networks (IPSN 2004)*, pages 71–80, New York, NY, USA, 2004. ACM.

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

[156] Karim Seada, Marco Zuniga, Ahmed Helmy, and Bhaskar Krishnamachari. Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks. In *Second International Conference on Embedded networked sensor systems (SenSys 2004)*, pages 108–121, New York, NY, USA, 2004. ACM.

[157] T.S. Seidel, S.Y. Rappaport. 914 MHz path loss prediction models for indoor wireless communications in multifloored buildings. *IEEE Transactions on Antennas and Propagation*, 40:207–217, 1992.

[158] Roland Siegwart and Illah R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. Bradford Company, Scituate, MA, USA, 2004.

[159] Reid G. Simmons, David Apfelbaum, Wolfram Burgard, Dieter Fox, Mark Moors, Sebastian Thrun, and Håkan L. S. Younes. Coordination for multi-robot exploration and mapping. In *Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 852–858. AAAI Press, 2000.

[160] Raghupathy Sivakumar, Prasun Sinha, and Vaduvur Bharghavan. Cedar: a core-extraction distributed ad hoc routing algorithm. *IEEE Journal on Selected Areas in Communications*, 17:1454–1465, August 1999.

[161] Ivan Stojmenović. *Location Updates for Efficient Routing in Ad Hoc Networks*, pages 451–471. John Wiley & Sons, Inc., 2002.

[162] Ivan Stojmenovic and Xu Lin. Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks. *IEEE Transactions on Parallel Distributed Systems*, 12(10):1023–1032, October 2001.

[163] Ivan Stojmenovic and Xu Lin. Power-aware localized routing in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(11):1122–1133, November 2001.

[164] Ivan Stojmenovic, Mark Russell, and Bosko Vukojevic. Depth first search and location based localized routing and qos routing in wireless networks. In *International Conference on Parallel Processing*, pages 173–182, Washington, DC, USA, 2000. IEEE Computer Society.

[165] David J. T. Sumpter. *Collective Animal Behavior*. Princeton University Press, 2007.

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

[167] Andrew S. Tanenbaum. *Computer Networks (4. ed.)*. Prentice Hall Professional Technical Reference, 4th edition, 2002.

[168] Andry Tanoto, Jia Lei Du, Ulf Witkowski, and Ulrich Rückert. Teleworkbench: An analysis tool for multi-robotic experiments. In *First IFIP Conference on Biologically Inspired Cooperative Computing (BICC 06)*, pages 179–188, 2006.

[169] Héctor Tejeda, Edgar Chávez, Juan Sanchez, and Pedro Ruiz. Energy-efficient face routing on the virtual spanner. In Thomas Kunz and S. Ravi, editors, *Ad-Hoc, Mobile, and Wireless Networks*, volume 4104 of *Lecture Notes in Computer Science*, pages 101–113. Springer Berlin / Heidelberg, 2006.

[170] Onur Tekdas, Yang Wei, and Volkan Isler. Robotic routers: Algorithms and implementation. *International Journal of Robotics Research*, 29:110–126, January 2010.

[171] Sebastian Thrun. Particle filters in robotics. In *Seventeenth Annual Conference on Uncertainty in AI (UAI)*, 2002.

[172] Sebastian Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*, pages 1–35. Morgan Kaufmann Inc., San Francisco, CA, USA, 2003.

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

[174] E. Ugur, A.E. Turgut, and E. Sahin. Dispersion of a swarm of robots based on realistic wireless intensity signals. In *Twenty Second International symposium on Computer and Information sciences (ISCIS 2007)*, pages 1–6, November 2007.

[175] D. M. Smith W. N. Venables and the R Development Core Team. An introduction to r. http://www.r-project.org/, March 2012. Accessed on July 1, 2012.

[176] I.A. Wagner, M. Lindenbaum, and A. M. Bruckstein. Distributed covering by ant-robots using evaporating traces. *IEEE Transactions on Robotics and Automation*, 15(5):918 –933, October 1999.

[177] Guiling Wang, Guohong Cao, and Thomas F. La Porta. Movement-assisted sensor deployment. *IEEE Transactions on Mobile Computing*, 5:640–652, June 2006.

[178] Guiling (Grace) Wang, Guohong Cao, Piotr Berman, and Thomas F. La Porta. Bidding protocols for deploying mobile sensors. *IEEE Transactions on Mobile Computing*, 6:563–576, May 2007.

[179] Thomas Williams, Colin Kelley, and many others. Gnuplot 4.4.4: an interactive plotting program. http://gnuplot.sourceforge.net/, November 2011. Accessed on July 1, 2012.

[180] Alan FT Winfield. Distributed sensing and data collection via broken ad hoc wireless connected networks of mobile robots. In *Fifth International Symposium on Distributed Autonomous Robotic Systems (DARS 2000)*, pages 273–282. Springer, October 2000.

[181] Brian Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the Second International Conference on Autonomous agents (AGENTS 1998)*, pages 47–53, New York, NY, USA, 1998. ACM.

[182] Mohamed Younis and Kemal Akkaya. Strategies and techniques for node placement in wireless sensor networks: A survey. *Ad Hoc Netw.*, 6(4):621–655, June 2008.

[183] Marco Zúñiga Zamalloa and Bhaskar Krishnamachari. An analysis of unreliability and asymmetry in low-power wireless links. *ACM Transactions on Sensor Networks (TOSN)*, 3(2), June 2007.

[184] Xiaoming Zheng, Sonal Jain, S. Koenig, and D. Kempe. Multi-robot forest coverage. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, pages 3852 – 3857, August 2005.

[185] Michele Zorzi and Ramesh R. Rao. Geographic random forwarding (geraf) for ad hoc and sensor networks: Multihop performance. *IEEE Transactions on Mobile Computing*, 2(4):337–348, October 2003.

[186] Le Zou, Mi Lu, and Zixiang Xiong. Pager: a distributed algorithm for the dead-end problem of location-based routing in sensor networks. In *International Conference on Computer Communications and Networks (ICCCN 2004)*, pages 509–514, October 2004.

[187] Y. Zou and Krishnendu Chakrabarty. Sensor deployment and target localization based on virtual forces. In *Twenty-Second Annual Joint Conference*

of the IEEE Computer and Communications (INFOCOM 2003), volume 2, pages 1293 – 1303, March-April 2003.