NETWORK ISSUES FOR 3D WIRELESS SENSOR NETWORKS

By

Fernando J. Cintrón

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science — Doctor of Philosophy

2013

ABSTRACT

NETWORK ISSUES FOR 3D WIRELESS SENSOR NETWORKS

By

Fernando J. Cintrón

Wireless sensor networks (WSN) give the opportunity to monitor the environment by performing sensing tasks in places that are difficult to reach or dangerous for humans. Nevertheless, topographical characteristics of such places and the sensor node's limitations introduce new issues in WSN performance. Additionally, in scenarios where sensors are moving or in rugged terrain, there is a high chance for them to be out of communication range, causing network connectivity problems. Hence, solutions have to take into consideration the aspect of the topography consisting of its three dimensional characteristics, namely, type of terrain, terrain unevenness, and obstacles.

This dissertation discusses several research topics addressing issues relevant to WSN connectivity and area coverage problems. First, changes in sensor communication range are studied by varying sensors' heights relative to the surface. A novel communication technique that relies on the jumping capabilities of sensors is proposed. While the jumping sensor robots are airborne, the change in elevation enhances their ability for a short time to successfully communicate with other sensors that are out of communication range at the ground level. Field experiments were conducted and results show a considerable improvement in wireless communication ranges.

Second, the impact of network connectivity and area coverage in a jumping sensor network is further studied. A Hopping Sensor Network Model is defined to increase sensing area coverage along with the enhancement of network connectivity. A Hopping Sensor Rout-

ing Protocol is designed from the model that balances the energy consumption on active jumping sensor nodes. Results from simulations show the increase in area coverage obtained from jumping sensor networks, and the effectiveness of the routing protocol to optimize communication paths while balancing energy depletion in the network.

Third, a distributed wireless sensor network organization to establish a functional network, without requiring initial topology information, is presented. Two decentralized algorithms that use the jumping capabilities of sensors are designed for the discovery of isolated sensors. Simulation results show the success of the algorithms to enhance base station reachability. Additionally, cluster to cluster (C-to-C) packet forwarding schemes relying on boundary jumping sensor gateways are defined and analyzed, showing remarkable savings in network energy consumption.

Fourth, in order to have a functional network, it is important to address connectivity issues in an application oriented manner. This work presents an efficient node redeployment-decision process to produce a functionally heterogeneous (jumping and non-jumping sensors) WSN with a performance guarantee. Network performance is defined as a network fitness formula considering the network Quality of Connectivity (QoC). Decision making algorithms for node relocation and topology defragmentation are presented, along with a discussion of their performance.

Fifth, a multi-step procedure to produce a direction oriented jumping sensor network is presented. A jumping sensor robot approach is introduced for collecting and processing signal strength data into relative geographical orientation information. A directional-orientation decision algorithm is defined to process the orientation information. Furthermore, an error identification and correction procedure is established. This has proven to accurately fix the true orientation of the nodes by using only a pair of location aware beacon nodes.

To my family.

ACKNOWLEDGMENTS

Pursuing a Ph.D. degree requires a high level of commitment and perseverance. Completing this dissertation has been possible thanks to the guidance and support of many people. I would like to thank everyone who has contributed directly or indirectly toward this accomplishment, marking one big step in my professional development.

I would like to express my sincere gratitude to Dr. Matt W. Mutka for his advices, financial support, and incessant patience. As a professor, he introduced me to a new area of research in wireless sensor networks. His experience and guidance helped me to become a better researcher and to successfully complete this dissertation.

I would like to thank Dr. Li Xiao, Dr. Abdol-Hossein Esfahanian, and Dr. Ning Xi for their valuable comments in this dissertation and for taking the time to serve on my Ph.D. guidance committee. I am also thankful to the Department of Computer Science and Engineering and the Graduate School of Michigan State University for providing me teaching assistantships, and awarding me a fellowship to support this research.

I would like to express my thanks and deepest appreciation to Dr. Percy Pierre, for his generosity, support, and his role with the Sloan program, giving me the opportunity to visit and meet so many great people at Michigan State University before making my decision to study in this institution. Without any doubt, the Alfred P. Sloan foundation program played an important role from the beginning, deciding on an academic institution to pursue the Ph.D. degree; while on board, giving me the support and encouraging me to stay focused; to the end, becoming part of such a great network of people with diverse backgrounds.

I would like to thank my colleagues in the ELANS laboratory. We spent time having discussions that resulted in great research collaborations.

Last but not least, I would like to thank my family, who encouraged me to continue on this journey and to always keep growing as a person. I would like to express my deepest thanks to my wife Maria Teresa, for her unconditional support, patience and love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

Chapter 1

INTRODUCTION

Wireless Sensor Networks (WSNs) serve a broad range of applications, for e.g., environmental monitoring that includes sensing of temperature, sound, movement, earth tectonic plate vibrations, pressure, and toxic condition among others. WSNs offer the ability to monitor large scale areas where wired networks may be unviable or infeasible to establish. Furthermore, they present the opportunity to monitor the environment by performing sensing tasks in places that are difficult to reach or dangerous for humans. Nevertheless, topographical characteristics of such places and the sensor node's limitations introduce new issues in WSN performance. Additionally, in scenarios where sensors are moving or in a rugged terrain, there is a high chance for them to be out of communication range, causing network connectivity problems.

This dissertation presents solutions to improve WSN connectivity and network deployment efficiency. It considers sensor nodes' limitations in combination with three-dimensional characteristics of the topography, namely, type of terrain, terrain unevenness, and obstacles. With this goal present, novel techniques optimized for sensor nodes with limited capabilities are presented. In this chapter, challenges of today's WSN are introduced, along with motivations for connectivity efficient and enduring networks.

1.1 WSN Challenges

Although WSNs are self-organizing and provide an ease of deployment, the sensor nodes have inherent shortcomings in form of limited battery, processing capabilities, and communication range.

1.1.1 Connectivity and Area coverage

Sensor nodes used in the network have the task to monitor their surrounding environment. The area that they cover depends on two factors: (1) the sensing range of the sensors, and (2) the connectivity of the sensors. The sensing range is a property dependent on the type of sensing, which is defined by the application's demands. In other words, it can be considered as an application-logistic problem. For the scope of this dissertation, connectivity is defined as follows: a single node is said to have connectivity if it has a direct communication path to the base station (one communication hop) or an indirect communication path to the base station via its neighbor nodes (multi-hop routing communication). In contrast to the sensing range, the connectivity presents a dynamic problem for mobile sensor networks and topography scenarios requiring less precise deployment techniques.

Network Connectivity is an important factor because when a sensor is not able to reach the base station (i.e., it does not have connectivity), the data gathered by the sensor cannot be transferred to the base station. Depending on the application's tolerance to communication delay and unpredictability of events for the disconnected sensor to regain connectivity, the collected data can become outdated. Therefore, connectivity is a crucial aspect for the network to operate.

1.1.2 Wireless Communication Challenges

Wireless communication is affected by many factors: environment, distance, antenna type and height, etc. These factors can cause signal fading, resulting in reduced communication range, which can negatively affect the transmission of network packets.

The distance between sensor nodes is one of the critical factors that dictates the received signal strength (RSS). The farther away the sensors are from each other, the weaker the received signal strength will be, and the tendency for packet loss increases. Ideally, all sensors would be placed within the wireless communication range of each neighbor. However, as previously stated, this becomes infeasible in scenarios where sensors are moving, or in difficult-to-reach terrains where more aggressive deployment techniques (e.g., air-borne dispersion) are required. In such cases, the deployment of sensors becomes unevenly spread over the target field. Furthermore, as more aggressive the deployment technique becomes, the impact on the design of the sensor, e.g., usage of short length antennas, creates a further reduction of communication range.

Beside the limitation of sensor nodes, sensors' specification often advertise communication characteristics under ideal conditions, not reflecting the adversity of the conditions presented to today's use of sensor networks. For instance, sensor networks are deployed in rough terrains where ground unevenness and obstacles greatly interfere with wireless communication. Moreover, sensor nodes tend to be small in size and height, making heavy foliage a potential line of sight obstacle in wireless communication.

Given the nature of sensor network deployment and the constraining factors, it is almost certain for a WSN to experience connectivity related problems.

1.2 Motivation

1.2.1 Communication Range Increase in WSN

It is known that a change in antenna elevation affects wireless communication range. The communication range of a sensor node increases with its elevation with respect to ground level, within practical bounds.[1] A temporary change in sensor elevation is seen in jumping sensor robots. While the sensor is jumping, the change in elevation enhances, for a short time, its ability to communicate with other sensors that were not reachable at ground level.

The jumping sensor (also known as hopping sensor for its movement resemblance to grasshoppers) concept is inspired by natural wild life behaviors to perform sensing tasks in environments where it becomes difficult for other types of sensor robots, e.g., wheeled sensors. The jumping motion is not as accurate as the wheeled motion, but is adequate for rough terrains where wheeled sensors cannot perform as desired. Figure 1.1 shows three jumping sensors[2] at ground level, while one jumping sensor is in the act of jumping (airborne). The picture was enhanced to display a close view of the jumping sensor. Dotted red lines highlight the airborne sensor and the embedded magnified view.

Jumping sensors provide an alternative to address common WSN's problems. The use of jumping sensors requires the study of three-dimensional aspect of network topologies. However, WSN are usually modeled in a two-dimensional space, leaving out the effects of topology characteristics issues, and dynamic changes in sensor nodes' height with respect to others.

[1] Communication impact from sensor node elevation is presented in Chapter 2.

[2] Jumping sensor prototypes presented in Fig. 1.1 are developed by Zhao *et al.* (2009).

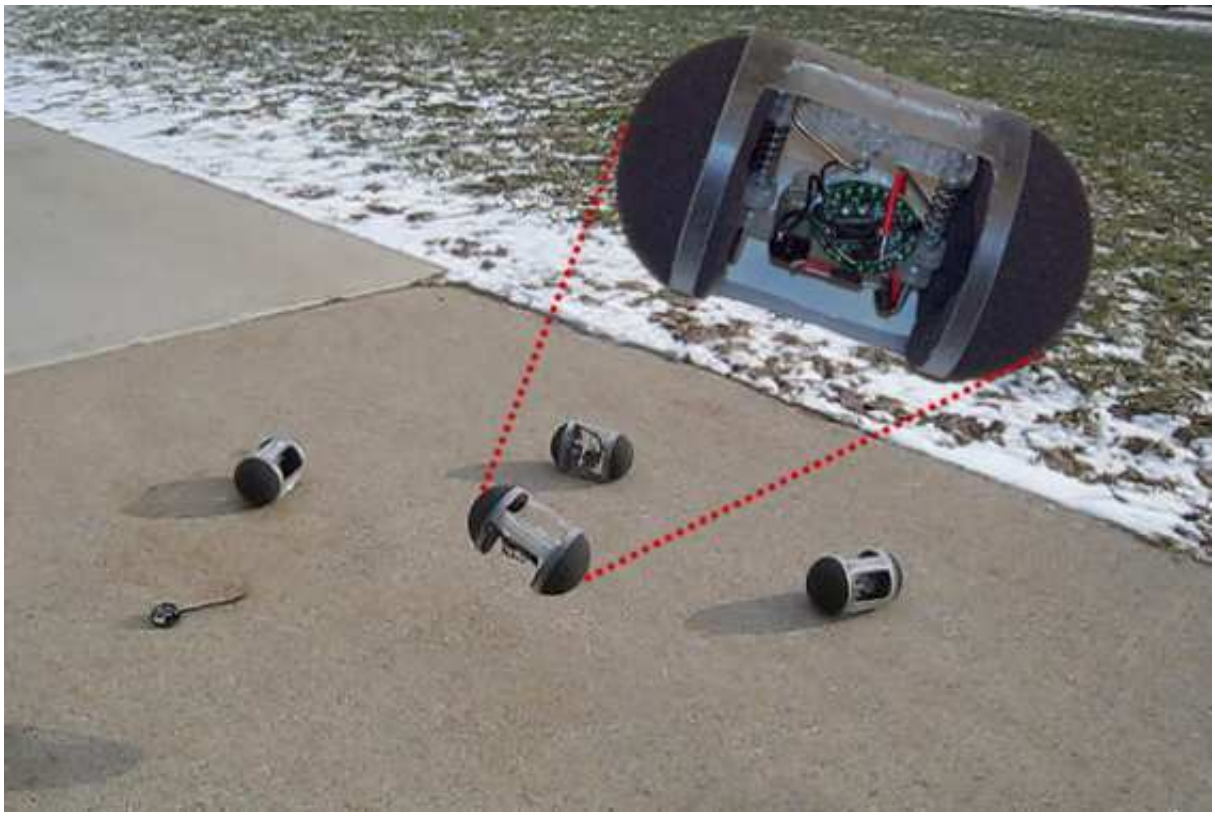


Figure 1.1: Active Airborne Jumping Sensor, Zhao *et al.* (2009). For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this dissertation.

1.2.2 Jumping Sensor Networks and Connectivity

When wireless sensor nodes are randomly deployed in a field, there may be areas that lack sensor nodes. These areas tend to create *communication holes*, and in the worst case, break the topology into disconnected clusters of nodes, i.e., disconnected groups of interconnected nodes.

The problem of communication holes in WSN can be addressed using different approaches. An intuitive method is to increase the node density, i.e., amount of nodes per unit area. This approach provides a feasible solution for tasks, which can be performed with relatively cheap technology. However, leading edge technology demanding applications are expensive, given that the price per node tends to be higher. On the other hand, node relocation provides the

opportunity to use those nodes that are redundant in some areas to cover communication holes. However, relocation is not possible in difficult to navigate terrains such as river stream, lakes, or harmful areas. As an alternative to reach farther, signal transmission power can be amplified. Nevertheless, many *topology control protocols* for WSN use maximum transmission power of the nodes at their initial phases, and still experience connectivity problems, due to the presence of communication holes.

Complete and persistent network connectivity within a random wireless sensor network topology is difficult to guarantee. Instead, jumping sensors can be used to temporarily increase connectivity in wireless sensor networks. However, a topology of jumping sensor nodes configured to enhance network connectivity by means of jumping can significantly reduce the network life span, if not well managed. With the help of selected jumping sensors on disconnected clusters, enhanced connectivity can be achieved without fully compromising network energy.

1.2.3 Network Fitness

In order to have a functional network it is important to address connectivity issues in an application oriented manner. However, efficient node redeployment is a key aspect for enduring WSN.

WSN applications require a certain level of network performance guarantee to properly function. Some applications can withstand an intermittent connectivity (delay tolerant), meaning that communications between node pairs do not require long lasting wireless communication links. Instead, the communication occurs for short periods of time during the network lifespan. In such cases, network cost for redeployment could far exceed the gains provided by jumping sensors. It is clear that connectivity alone is not representative of

potential network performance. Therefore, an evaluation metric representing the fitness of a WSN in terms of the quality of its connectivity is required.

Part of this research is dedicated to providing a Quality of Connectivity (QoC) performance metric for WSN. Furthermore, the contribution of having such metric is twofold. First, it provides a non-heuristic performance metric for the network topology, allowing almost instant topology evaluation after deployment. Second, it serves as an optimization metric for node redeployment.

1.2.4 Relative Orientation

Relative orientation is used by humans on a daily basis, as a set of directions to reach from one place to another, without knowledge of detailed location information. Many times, the orientation is obtained in the simplest form as place reference, guiding ourselves in a reference relaying manner, e.g., "Let's meet in the coffee shop north of the campus library.". While the orientation concept seems relatively simple in nature, it is still a challenge to purely obtain orientation information without relying on precise localization systems, sometimes impractical for a highly resource constrained (CPU, power, form-factor, etc.) WSN.

It is clear that as newer generations of jumping sensors become available, opportunities open up to solve many existing Wireless Sensor Network (WSN) problems, which include a node's relative orientation. We define a node's relative orientation as the ability of a node to identify the direction of a target (e.g., neighbor node) from its current location relative to a reference direction. Obtaining a reference direction, such as the magnetic north, has become simple with the usage of inexpensive magnetic compass sensors. However, the challenge remains to determine the relative direction of a neighbor node.

Jumping sensors display an opportunity to utilize the resulting airborne rotation, ob-

tained from the jumping action, to collaboratively identify neighboring nodes' relative position. This can be achieved with the usage of signal strength readings at different orientation angles. Such a method, presents a series of challenges including signal strength analysis, prediction and error correction methodologies.

1.3 Structure of the Content

This dissertation is organized as follow. Chapter 2 gives a background survey. Chapter 3 discusses wireless signal propagation from a WSN perspective. Fundamental factors responsible for decrease in successful wireless communications are presented and evaluated. Furthermore, it introduces an approach to decrease packet loss and increase communication range in a wireless jumping sensor network. Chapter 4 shows the impact of jumping sensors on connectivity and area covered, and presents a Hopping Sensor Network Model to increase the sensing area coverage. A hopping sensor network topology, and the introduction of two algorithms for node discovery and connectivity increase with the use of jumping sensors as gateways are discussed in Chapter 5. Chapter 6 presents an efficient node redeployment-decision process to produce a functionally heterogeneous (jumping and non-jumping sensors) WSN with a performance guarantee. Network performance is defined as a network fitness formula considering the network Quality of Connectivity (QoC). Chapter 7 a multi-step procedure to produce a direction oriented jumping sensor network topology is presented. Finally, Chapter 8 concludes this dissertation with an overall summary, and highlight areas for future research work.

Chapter 2

BACKGROUND SURVEY

This chapter provides a survey of related work to wireless sensor networks. It first presents existing work in the area of wireless sensor design. Second, since energy is at premium in sensor nodes, related work to efficient network energy management protocols is discussed. Third, connectivity and area coverage are topics directly addressed in this dissertation, therefore, work relevant to these topics is covered. Forth, topology control methodology is reviewed. Fifth, one decade of research progress in the area of jumping robots prototypes is presented. Sixth, a survey of state of the art sensors is provided. Last, related work to sensor localization and its challenges is discussed.

2.1 Wireless Sensor Design

The design of a wireless sensor network is a non trivial process. Römer and Mattern (2004), provided insight to different dimensions of the design space of a wireless sensor network. Some of the design aspect discussed are:

- Deployment: The deployment of sensor nodes into a target field is the initial phase to build a sensor network. However, it may be a continuous process during the life spam of the network, e.g., failed node replacement. Nodes may be installed at precise locations or deployed randomly. The deployment technique affects network properties such as node density, node locations, coverage, and connectivity.

- Mobility: Sensor nodes may have automotive capabilities, enabling to relocate them to target destinations. However, mobility may occur in a passive (incidental) manner, as the result from environmental influences such as wind, water streams, or carried by a mobile entity.

- Resources and Energy: The target application and the field environment dictate the type of sensors that the wireless network will be composed of, e.g., hundreds of small sensors vs. few sophisticated sensors. As sensor nodes become more sophisticated, the cost of a single unit becomes higher, which can be suitable for networks where few nodes are needed, but economically impractical for large-scale networks. Furthermore, the size of a sensor node can constraint available energy, computing, storage, and communication resources.

- Heterogeneity: Sensor networks may be composed of different sensor nodes and devices. Some nodes with special capabilities, such as, more computing power, greater storage resource, or longer communication range may be positioned on key locations to collect, process and route data from other sensors.

- Infrastructure: Wireless networks are built as infrastructure-based networks, ad-hoc networks, or a combination of both. In an infrastructure-based network, a set of base stations devices are deployed, to be used by sensor nodes to communicate. Base stations serves as intermediary relays on any communication. On the other hand, in an ad-hoc network, nodes directly communicate with each other in a collaborative manner, i.e., nodes can serve as routers for those nodes that cannot directly communicate, without the use of an intermediary base station.

- Network Topology: The maximum number of communication hops between any two

nodes in a network is known as the network diameter. This property is important since it affects network latency, and tolerance to node failure.

2.2 Network Energy Management

Among all the design aspects, energy and communication constraints have the highest influence on the node's performance. Wireless sensor life is usually limited by a battery source, hence it is important to reduce the energy consumption to maximize sensor life. There has been a significant amount of work carried out to reduce the impact of energy usage for wireless sensor operation.

Shah and Rabaey (2002), proposed an energy aware routing scheme that uses sub-optimal paths occasionally to provide substantial gains. Their approach is based on the premise that always using lowest energy paths may not be optimal from the point of view of network lifetime and long-term connectivity. An uncontrolled usage of lowest energy paths may deplete the energy on nodes located on critical routes, creating a disruption in network connectivity.

Data-gathering is an important issue in wireless sensor network since energy is usually a scarce resource on the sensor nodes. The problem of energy efficient data-gathering escalates when nodes are capable of mobility. Liu *et al.* (2004); Liu and Lee (2004) provided clustering-based and time-driven protocols to minimize the energy consumption for data-gathering in wireless mobile sensor networks.

Carbunar *et al.* (2006), focused on the detection and elimination of redundant sensors without affecting network coverage, improving energy efficiency on the network. Similarly, an approach was introduced by Cheng and Yen (2006), to obtain an optimal active sensor

selection while retaining system coverage. Likewise, Ding *et al.* (2007), presented a connectivity based Partition Approach (CPA) to partition sensors into groups. Groups follow a sleep scheduling for sensor nodes, preserving network communication quality and reducing the energy consumption on the sensor network. CPA differs from other approaches since it is based on measured connectivity, and not on nodes' locations.

2.3 Connectivity and Area Coverage

In wireless communication signal strength fades over distance. Low signal strength negatively contributes to high packet loss, increasing the possibility of communication links breakage. Aguayo *et al.* (2004), analyzed the importance of network planning over packet loss and the implications for MAC and routing protocol design. In order to better understand wireless network performance, signal strength estimation for indoor communications was performed by Biaz *et al.* (2005). Similarly, Kavak *et al.* (1999) studied the effects of base station antenna height and mobile node movement over wireless communication.

Maintaining system connectivity in a wireless sensor network is a key issue for maximizing area coverage. Wang *et al.* (2003), studied the relation between wireless sensor network coverage and connectivity. In fact, they observed that coverage infers connectivity if the radio transmission range is at least twice the sensing range, i.e. $R_t \geq 2R_s$. The relation allowed them to treat coverage and connectivity as one problem. Similarly, Zhang and Hou (2005), obtained the same coverage and connectivity relation ($R_t \geq 2R_s$), proving that when the relation is satisfied, complete coverage of a convex area implies connectivity among the set of nodes in that area.

Clustering techniques have been discussed to achieve higher network connectivity. Younis

et al. (2006) studied two clustering methods, for wireless sensor networks. In the first method, multiple gateways are deployed, and sensor nodes associate with them to form clusters. The second method is a multi-tier architecture, in where selected sensors are designated as agents to connect unreachable nodes to a single gateway. Multi-tier clustering showed superior node reachability, but requires more sensors involved in network management.

In mobile wireless sensors, area coverage is often achieved by relocating sensors to needed areas. Howard *et al.* (2002), used virtual forces to redistribute sensors. Sensors are repelled from obstacles objects or from others nodes in order to maximized the covered area. Similarly, Zou and Chakrabarty (2003) proposed a Virtual Force Algorithm (VFA) to enhance sensor field coverage. However, even with groundbreaking relocation techniques there will be situations where the relocation of the sensors to needed areas is not possible due to obstacles or terrain types (eg., river streams, lakes, etc.). This become a major problem when sensors are sparsely distributed over the field (e.g., when using air deployment), because there is a higher possibility for sensors to be left without connectivity, consequently affecting area coverage.

2.4 Topology Control

The topology of an ad hoc network is defined as the set of communication links between nodes pairs used explicitly or implicitly by a routing mechanism, Ramanathan and Rosales-Hain (2000). Different factors can affect a topology. Some of them can be classified as uncontrollable such as node unpredicted movement, noise interference, and obstacles. Others can be controllable such as node transmission power, node relocation, and by limiting the amounts of neighbors that each node keeps track.

Topology control is important since wrong topology control will result in degraded performance and even poor connectivity on the network. For instance, a too dense topology will limit the spatial reuse, hence network capacity is reduced. On the contrary, if the topology is too sparse, connectivity can be degraded. Moreover, a good power control is necessary to extend battery life of nodes, and to avoid extremes power levels. In fact, too high transmit power level result in excessive interference among nodes, creating high MAC-level contentions. On the other hand, too low transmit power level results in a disconnected network.

The following methods address the problem of controlling the network topology by adjusting the above mentioned controllable factors.

2.4.1 Degree-based Mobility Model

Mobility of nodes in wireless network introduces more complexity in the system models. There are some studies in literature that propose mobility models in order to decrease mobility effect on network performance and permit to test algorithms for realistic scenarios. Mobility models can represent mobile nodes whose movements are independent of each other, or mobile nodes whose movements are dependent on each other Camp *et al.* (2002). Some well-known mobility models are: random walk, random waypoint, random direction, Gauss-Markov and parallel path, they differ from each other about the choice of speed and direction of the nodes.

- Random walk: a simple mobility model based on random directions and speeds.
- Random waypoint: a model that includes pause times between changes in destination and speed.

- Random direction: a model that forces mobility nodes to travel to the edge of the simulation area before changing direction and speed.

- Gauss-Markov: a model that uses one tuning parameter to vary the degree of randomness in the mobility pattern.

- Parallel path: a model where a mobile node picks a random speed and move across the geographical area following a direction parallel to one of the boundary lines.

In Santi (2005), a different approach of mobility model is proposed: mobility is used to achieve a physical topology with better performance in connectivity, instead of decrease its impact. At this aim, mobility is created in an artificial way: a static wireless network is overlaid by a mobile wireless network. The static network is assumed as a wireless sensor network or a wireless mesh network (where nodes take data and send it to the sink or to the backbone) and the mobile network is constituted by nodes that follow a certain mobility pattern. Direction of the mobile nodes is determinate by a measure of importance of static neighbors. In particular, a degree-based mobility model is defined, where the probability distribution for the direction depends on the degree of the static nodes.

In its work, topology is generated in the following manner: a sink node is placed in the central geographic area and static nodes are placed around it in a growing manner, where new static nodes are placed around previously placed static nodes, to avoid isolated nodes and to have a fully-connected network. Every static node is ensured to be in the coverage area of at least one and at most k neighbors, where k is a chosen parameter accordingly to the desired coverage. A multi-hop connectivity is considered between nodes and the sink, when data is sent to the sink.

The speed of mobile nodes is uniformly distributed between $[0, V_m]$ and the probability distribution of direction is proportional to the importance of the neighboring nodes. Hence, mobile nodes move through more important static nodes. In particular, in Yanmaz (2008), the distribution of the direction of a mobile node depends on the degree of its neighbors, where degree of a node is defined as the number of connections of that node with nodes that are within its transmission range.

This mobility model increases the connectivity of a sparsely-connected grid topology and the random topology. Moreover, using mobile nodes overlaying a static network decreases the average static node-to-sink number of hops, creating shortcuts between nodes of the static network.

2.4.2 Transmit Power Adjustment

Topology control is considered in Ramanathan and Rosales-Hain (2000), as a constrained optimization problem of practical importance; particularly as minimizing transmit power subject to two constraints: network connected or network biconnected. Therefore the problem in this case is adjusting the transmit powers of nodes to obtain a desired topology.

There are two centralized algorithms in order to minimize the maximum power used per node for use in static networks: CONNECT and BICONN-AUGMENT. For mobile networks two distributed heuristics are introduced, Local Information Non Topology (LINT) and Local Information Link-State Topology (LILT). They adapt node transmit powers related to topological changes and aim to have a connected topology using minimum power.

CONNECT is a simple algorithm where every component is merged iteratively until one component is left. In the BICONN-AUGMENT a connected network is increased to obtain a biconnected network, which is based on CONNECT. In both cases a post processing can

be used to delete redundant connections and to ensure per-node minimally.

In LINT, each node has three parameters: a desired node degree, a high and a low threshold on the node degree. Periodically, each node changes its power level with respect to the number of its active neighbors, keeping the node degree between the two thresholds. LILT is an improvement of LINT, it overrides the high threshold when there is a topology change from the routing protocol update results (i.e. link-state protocols). It is composed of two main parts:

- Neighbor Reduction Protocol (NRP): In charge of maintaining the node connectivity degree around a certain configured value.
- Neighbor Addition Protocol (NAP): Employed to increase the node transmit power if the topology changes indicated by the routing update is in undesirable connectivity.

Initially all nodes start with the maximum possible power to obtain the maximum connected network, enabling successful propagation of updates and the initialization of a network topology database at each node. After initialization, a node receiving a routing update has to determine in the state of the updated topology: *disconnected*, *connected (not bi-connected)*, or *bi-connected*. NRP and NAP are activated on each node, depending on the topology state. If the topology is disconnected, the node increases the transmit power to the maximum possible value. If connected (not bi-connected), special bi-connectivity augmentation is performed, in where power changes are coordinated with other nodes to avoid all nodes to react to the topology changes. Finally, if the topology is found to be bi-connected, no action is taken.

It is important to note that node coordination is achieved by setting a controlled randomized timer for each node, before taking an action. Nodes proceed to take an action only

if the topology is still in an undesired state after the timer has expired.

2.4.3 Local Minimum Spanning Tree (LMST)

LMST, described in Li *et al.* (2003), is a minimum spanning tree (MST) based topology control algorithm in where each node builds his local minimum spanning tree independently and only keeps as neighbors those nodes that are one-hop away from it. The topology constructed under MST preserves the network connectivity and bounds by 6 the degree of any node. A small node degree is desired because it reduces the MAC-level contention and interference among nodes. Moreover the resulting topology can be further converted in one with bi-directional links. Having a topology that only consist of bi-directional links is important for link level acknowledgments, and medium access control mechanisms such as RTS/CTS in IEEE 802.11.

Besides the algorithm presented in Li *et al.* (2003), guidelines to design an effective topology control are introduced:

- Network connectivity has to be preserved with the use of minimal possible transmission power on every node.
- Algorithm has to be distributed: each node has to make decision based on the information it has collected from the network.
- Algorithm has to be dependent only from locally information, to have less impact from mobility.
- Bi-directional links and small node degree are desired on the topology.

Moreover, the network topology constructed under the common maximum transmission

range for all nodes, denoted as d_{max}, is represented as a graph $G = (V, E)$, where V is the set of the nodes in the network and E is the edges set of G, which represents connection links between nodes that are within the transmission range d_{max}. Each node (u) is assigned with a unique id and it is defined to have a visible neighborhood $NV_u(G)$. The visible neighborhood $NV_u(G)$ is the list of neighbor nodes that node u can reach by employing its maximum transmission power.

LMST algorithm has three phases: information collection, topology construction and determination of transmission power; with an optional optimization phase for the construction of the topology with bi-directional edges.

The **first phase**, namely information collection, is the exchange of information by each node. This information is composed by the node id and position. This aim can be reached using a periodic message transmitted with the common node maximum transmit power. The time interval between two of these messages depends on the level of node mobility in the network.

In the **second phase**, topology construction, each node applies Prim's Algorithm independently to obtain its local minimum spanning tree. Two points are fundamental to have a power efficient minimum spanning tree: the weight of an edge has to be the transmission power between the two nodes, and the local minimum spanning tree constructed by each node has to be unique. Since multiple edges with the same weight exist, the minimum spanning tree is not unique. Therefore, a new weight function is defined to guaranteed that in each step of Prim's algorithm the choice of minimum weight edges is unique, resulting in an unique local minimum spanning tree per node.

It is assumed the maximal transmission is known and it is the same to all nodes. With the message in the first phase every node knows the specific receiver power level of each

of its neighbors. In the **third phase**, the transmission power that a node needs to reach its neighbors is determined considering two propagation models: the free space propagation model and the two-ray ground propagation model. In both models, there is a common relation between P_t (power used to transmit) and P_r (power received):

$$P_r = P_t \cdot G \tag{2.1}$$

where G is a function of the antenna gains of the receiver and transmitter, the antenna height of the transmitter and receiver, the distance between transmitter and receiver, the wave length and the system loss. In other words, if we have two nodes A and B, when A receives the position information from B, it measures the receiving power P_r and obtains G for node B as:

$$G = \frac{P_r}{P_{max}} \tag{2.2}$$

Therefore, in order for B to be able to receive A's message, A needs to transmit at least:

$$P_{th} \cdot G \tag{2.3}$$

where P_{th} is the minimum power threshold of a node to successfully receive a message.

In the optional optimization phase, a bi-directional network is built with two solutions: to enforce all the uni-directional links to become bi-directional; or to remove the uni-directional links and place them with two new topologies. The first one gives more routing redundancy, while the second one is more efficient in terms of spatial reuse. Furthermore, different important properties of LMST algorithm are proved, and it is determined how often the neighbors information should be exchanged for the topology to be updated, by using a

probabilistic model of node movement.

2.5 Mobile Robots

The development of new kinds of mobile sensors that can deal with terrain adversities has always been a stimulating topic of research. Jumping robot technology, also known as hopping robots, are characterize by their jumping method of movement. Jumping robot development has improved during the last decade, but working prototypes are still scarce. Nevertheless, this technology represents an unique opportunity for wireless sensor networks to be deployed in hard to reach areas or difficult to navigate with traditional wheeled mobile robot sensors.

Wei *et al.* (2000), built a prototype no bigger than a $5cm$ cube size, powered by a battery source, with no active directional control, resulting in constrained movement. A more in depth design for a jumping robot, including a working prototype, was later presented and suggested for usage in celestial body exploration by Burdick and Fiorini (2003).

Increasing jumping height and horizontal displacement, with power source limitations has been one of the main are of research for this technology. Scarfogliero *et al.* (2007), developed a jumping robot inspired from frog locomotion to achieve longer jumps. Similarly, Kovac *et al.* (2008) presented a miniature prototype with only 7 $grams$ of mass, able to achieve jump heights of more than 1 m.

A combustion-driven piston prototype of a hopping robot was developed by Sandia National Laboratories[1]. The hopping robot can make jumps up to 3 ft (approx. 0.9 m) of height, and move horizontally up to 6 feet (approx. 1.8 m) (German (2000)). It is capable of

[1] `http://www.sandia.gov`

making $4,000$ jumps on a single fuel tank (20 grams of fuel). Another experimental hopping robot can make jumps up to 20 ft (approx. 6 m) of height, and can perform about 100 jumps on a single fuel tank.

As hopping robots' designs evolve, the opportunity to incorporate them into a wireless sensor network arises. Zhao *et al.* (2009), discussed the design and development of a manually loaded, self-stabilizing jumping robot that carries a microcontroller with sensing capabilities. It uses a 3.7 V, 100 mAh LiPo battery, and is capable to jump up to a height of 0.35 m. An enhanced design able to perform continuous jumps was later introduced, Zhao *et al.* (2010).

Research involving hopping sensors has been focused on their relocation. Cen and Mutka (2008), discussed hopping sensor relocation, considering the impact of air disturbance and movement inaccuracy. Similarly, Pei *et al.* (2009); Kim *et al.* (2010), provided relocation algorithms for hopping sensors in obstacle abundant terrains. Other relocation algorithms with balanced migration of hopping sensors, are presented by Kim and Mutka (2009a,b). Their objective is to avoid the creation of new sensing holes from repeated hopping sensors requests made to a single supplier.

2.6 State of the Art Sensors

There are different wireless sensor nodes available in the market nowadays. Many of them are characterize by their small size and low power consumption, Warneke *et al.* (2001). MICA2, MICA2DOT, and IRIS are some of the wireless measurement system modules developed by MEMSIC[2]. These modules come with expansion connectors for light, temperature, relative humidity, barometric pressure, acceleration and seismic activity, acoustic, magnetic field, and

[2] `http://www.memsic.com/wireless-sensor-networks/`

others sensor boards. The main differences are on the module size, battery types, data rates, transmission frequencies, and advertised communication ranges. Some of these specifications are summarized in Table 2.1.

It is worth to note that experimental communication range for these modules, on open field scenarios, is much smaller than the advertised. More details on experimental results is provided in Chapter 3.

Table 2.1: Wireless Measurements Systems Specifications

	MICA2	MICA2DOT	IRIS
Battery	2X AA	3V Coin Cell	2X AA
Size[3](mm)	58 x 32 x 7	25 x 6	58 x 32 x 7
Weight[3] (grams)	18	3	18
Radio Center Frequency	868/916 MHz, 433 MHz, 315 MHz		2.4 GHz
Data Rate (Kbps)	38.4		250
RF Max. Power (dBm)	5, 10		3
Receive Sensitivity (dBm)	-98, -101		-101
Outdoor Range (m)	150, 300		300

2.7 Wireless Node Localization and Relative Orientation

Global Position Systems (GPS) are perhaps the most popular localization technology available. GPS receivers are able to obtain global coordinates with an accuracy within 3 meters, and even less than a meter when combined with augmentation systems, Aeronautics and Space Engineering Board (1995). Nevertheless, the real time operation and high accuracy of GPS receivers comes with a high power consumption, making this technology infeasible for many wireless sensor networks mainly composed of nodes with constrained power. For

[3]Size and weight of the modules excludes the battery pack.

this reason fewer specialized GPS enabled nodes are sometimes deployed to act as beacons. However, this also comes with complications since in order to establish a coordinate system, a minimum of three non-collinear beacon nodes are required for two dimensions localization, and four non-coplanar beacons for three dimensions localization. Therefore, beacon placement must be planned; something that can become impractical for scenarios where sensors are deployed in remote areas, e.g., volcano monitoring by air-dropped sensors, Song *et al.* (2009).

In addition, node localization with beacon nodes requires each node to estimate the distances to a subset of beacons. Beacon distances from the node are used to determine nodes' location with trilateration. Receiver Signal Strength Indicator (RSSI) is perhaps the less invasive method to estimate the distance between two communicating nodes. Signal path loss models are used to predict the signal strength over distance, Goldsmith (2005); Whitehouse (2002). Hence, the RSSI of beacons nodes is collected by the nodes and used to predict the distances.

Other methods for more accurate distance estimation require additional hardware. Speakers and microphones are used for Time Difference of Arrival (TDoA), in where acoustic signals are used to measure the signal traveling time and compute the distance Balakrishnan *et al.* (2003). On the other hand, Angle of Arrival (AoA) uses an antenna array or microphone array to determine the direction of propagation of a signal Priyantha *et al.* (2001). The drawback of these methods is the added mass and bulkiness they pose to the sensor robot, making them impractical for most mobile nodes. Moreover, the tight dependency to precise beacon placement presents a challenge to the design phase of WSN.

Node orientation is usually derived from each nodes' location coordinates. Location aware sensors utilize their position coordinates to navigate and reach to target destinations. This

approach becomes unfeasible for the vast majority of sensor nodes with limited capabilities. Furthermore, most relocation operations could rely in partial relative-orientation information obtained from neighboring nodes.

In Chapter 7, a multi-step procedure to obtain sensors' relative orientation is presented. The procedure employs the equipment already available in jumping sensors robot prototypes without additional hardware that could compromise the design and performance of the sensor nodes.

Chapter 3

WIRELESS COMMUNICATION RANGE INCREASE

3.1 Preliminaries and Motivations

Wireless communication can be affected by many factors: environment, type and height of antenna, noise, out of range receivers, etc. These factors can cause signal fading, reducing effective communication range, and increasing the tendency for packet losses at receiving nodes. If the communication between two sensor nodes experiences packet losses, retransmission may be required for those packets. Furthermore, if the sensor nodes are to distant from each other, they can become excommunicated from the network, creating network partitioning. Solutions approaches to deal with network partitioning and connectivity increase are discussed in Chapter 5.

This chapter studies the changes in sensor communication range, by varying sensors' heights relative to the surface. In order to increase wireless communication range in sensor networks, a communication technique that relies on the jumping capabilities of jumping sensors is proposed. While the sensor is jumping, a change in the altitude enhances, for a short time, the ability to communicate with other sensors. Base on this, a novel airborne communication process is defined. Field experiments were conducted and results discussions are provided.

3.1.1 Chapter Organization

First, a discussion of wireless signal strength from a sensor network perspective is given in Section 3.2. Experimental evaluations are presented and compared to existing signal path loss propagation models. Section 3.3 introduces airborne communication concepts and implications. A jumping algorithm and the airborne communication process are defined for the management of data transmissions in multiple jumps and the synchronization of transmitter and receiver nodes, respectively. Section 3.4 presents experimental evaluations. A chapter summary is provided in Section 3.5.

3.2 Received Signal Strength

Received Signal Strength (RSS) is the measured power of the signal when received by a node. RSS serves as an indicator of quality of signal reception. Therefore, to study communication patterns, a discussion of factors impacting signal strength and their importance to a jumping sensor communication is given. Among the different factors that can affect signal strength, this work will focus on the effects of communication distance, sensor elevation, and transmission power.

3.2.1 Impact of communication distance on RSS

In order to evaluate the impact of communication distance on RSS, experiments were performed in open areas such as a parking lot and a football field with natural grass turf. IRIS Crossbow sensor nodes were programmed to transmit packets at regular intervals, at their maximum transmission power (3 dBm). An IRIS base station mounted on a sensor board was used to measure the RSS of the packets. Measurements were taken at different separa-

tion distances between the transmitter and receiver. Two packet transmission interval rates ($50\ msec, 20\ msec$) were used to see the effect of packet rate over RSS. The impact of packet rate over RSS is found to be negligible for these experiments. Hence we present results from just the 20 $msec$ data rate experiment. This is depicted in Figures 3.1 and 3.2. It is worth to mention that IRIS sensor can receive packets with a signal strength as low as $-101\ dBm$ (IRIS Sensitivity), while it can provide the RSS Indicator (RSSI) as low as $-91\ dBm$ (IRIS Minimum Precision), which means that any received packet with a signal strength between $[-91,-101]dBm$ will be registered with $-91\ dBm$.

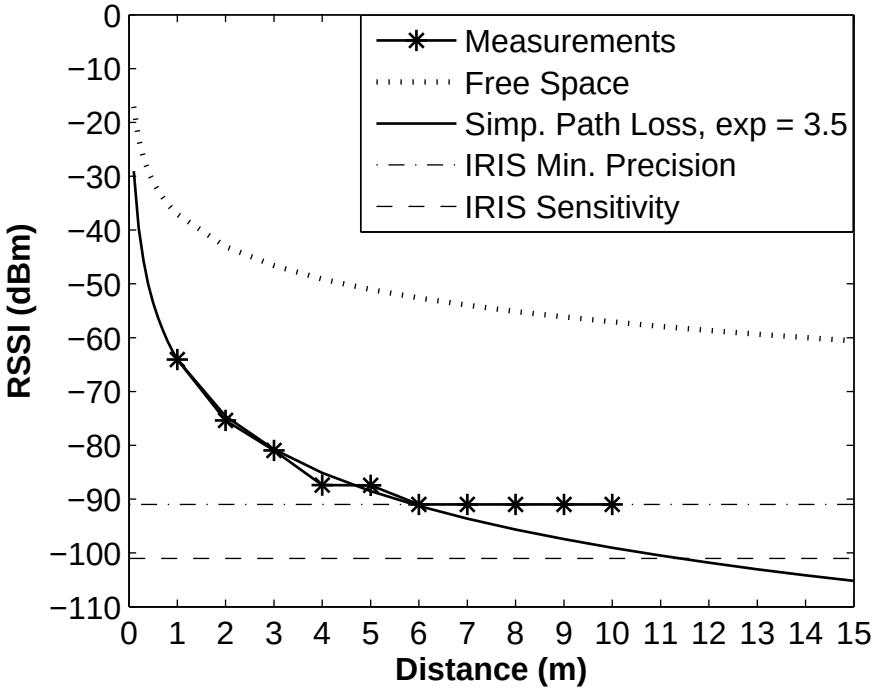


Figure 3.1: RSSI on Concrete

Figure 3.1 shows the impact of distance over RSSI, at surface level, in a concrete parking lot (flat surface). It can be seen to follow a simplified path loss model (with exponent 3.5) and accordingly the RSSI wanes to below measurable values by the time it reaches 10 m distance. Additionally, it was noticed that a significant amount of packet losses ($80 - 90\%$

losses) occurred at distances larger than 10 m between the sender and the receiver.

$$P_r = P_t K \left[\frac{d_0}{d}\right]^{\gamma} \tag{3.1}$$

The simplified path loss model is presented in Equation 3.1. P_r is the predicted received power when transmitting using power P_t from a distance d, and K is a constant (d_0/P_0) obtained from the received power P_0 at a reference distance d_0; γ is the path loss exponent (determined by measurements).

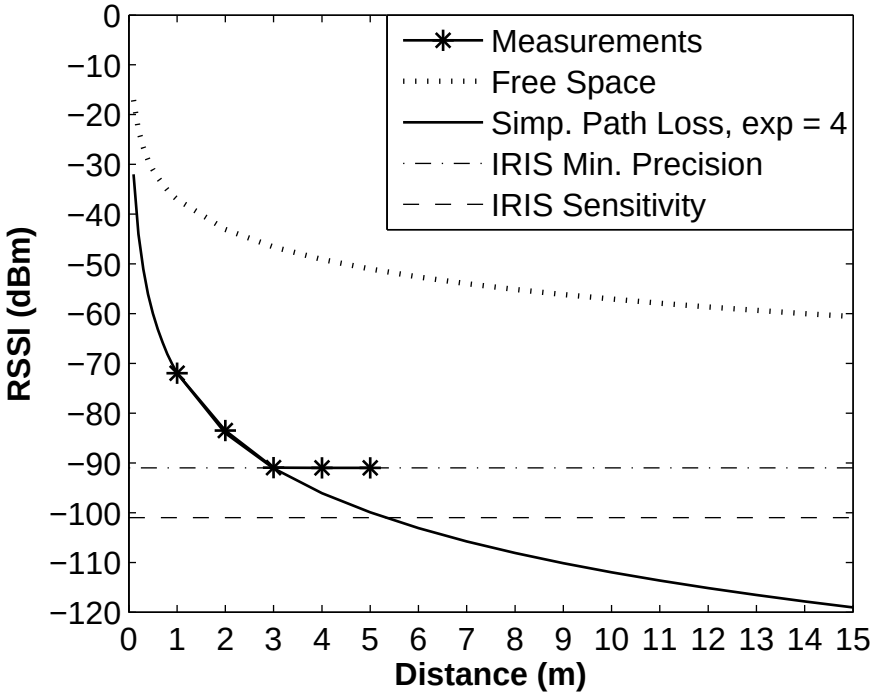


Figure 3.2: RSSI on Grass

Figure 3.2 shows the impact of distance over RSSI, at surface level, when measured in a grass turf. The average length of the grass was measured to be 4 cm long. It is worth to note that it follows again a simplified path loss model with adjusted γ value of 4.0, but the signal strength reduces to below measurable values at 5 m of distance. This happens because the thin strands of grass interfere with the signal, dissipating the signal strength, thus, reducing

the overall communication range.

3.2.2 Impact of sensor node elevation on RSS

Signal strength is affected by obstructions in the line of sight between the transmitter and receiver antennas. The Fresnel zone is an elliptical area surrounding the line of sight with foci collocated at the transmitter and the receiver, Green and Obaidat (2002). Obstructions within the inner 60% of the Fresnel zone, cause signal diffraction, which reduces signal strength.

Sensor nodes are often at ground surface level when randomly deployed. Therefore their transmitted signal strength gets attenuated faster with the increase in distance, resulting in the loss of all or part of the transmission packets. Experiments show that when the transmitting sensor can elevate itself, it has an increased opportunity of having successful communication with a distant receiver. This is attributed to the less interference from the ground surface irregularities to the Fresnel zone with the increase in altitude of the sensor node.

Experiments were performed to measure the impact of having the transmitter node at different heights (with respect to ground level) on communication range, with the receiver at ground level. For each height, packet transmissions were repeated for different distances between transmitter and receiver. The maximum communication range for each height was identified as the farthest distance having zero packet loss. Fig. 3.3 presents the communication ranges obtained for transmitter heights (0 to 1 m).

The obtained results clearly show that increasing the elevation (within practical bounds) of the sender, increases the communication range with zero packet losses. Furthermore, experimental results are compared with two signal path loss propagation models: Lee (1986),

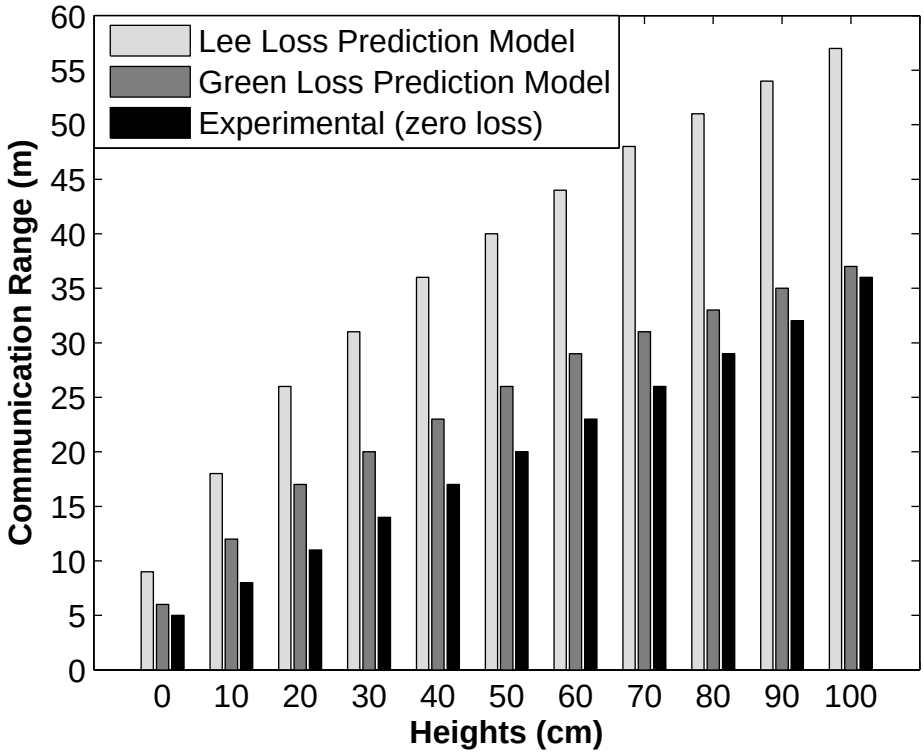


Figure 3.3: Communication Range with Varying Heights

Green and Obaidat (2002), Eqn. 3.2 and 3.3, respectively. Both models take into account gains from the transmitter and receiver antenna heights (h_t, h_r), while Green also considers loss due to the frequency of operation (f). Green's model appears to follow more closely to our experimental results as the height of the sensor varies from ground level to 1 meter, although the model was designed for heights above 1 meter.

$$P_{loss} = 40log_{10}(d) - 20log_{10}(h_t * h_r) \tag{3.2}$$

$$P_{loss} = 40log_{10}(d) - 20log_{10}(h_t * h_r) + 20log_{10}(f) \tag{3.3}$$

It is worth mentioning that most of the path loss models are designed for higher antenna

heights, longer communication distances, and lower frequencies of operation, e.g.: Hata path loss model, Hata (1980), for antenna heights over 30 m; Longley-Rice path loss model, Longley and Rice (1968), for distances over a Kilometer; Hata/Davidson path loss model, TIA TR8 Working Group 8.8 Technology Compatibility (1997), for frequencies up to 1500 MHz; Cost231-Hata path loss model, Mogensen *et al.* (1991), for frequencies up to 2000 MHz. Despite these incompatibilities, Lee and Green path loss models produced the closest prediction to our experimental results.

3.2.3 Impact of transmission power on RSS

The transmission power has a direct correlation with RSS and the transmission range. Though transmitting at lower transmission power saves energy, in a hopping sensor network, the energy saving is a small fraction of the entire transmission while hopping process. Hence, since communication range enhancement is the main focus of this work, it is advisable to always transmit at maximum transmission power.

3.3 Airborne Communication

This section introduces preliminary concepts for jumping sensor communication while airborne. A jumping algorithm is designed for the successful transmission of packets. An airborne two-way communication process is presented and evaluated with in-field experiments.

3.3.1 Time of Flight

The time of flight of a jumping sensor is the period of time it stays airborne. It is our goal to use this period of time to reach other sensors that may be distant and unreachable when staying at ground level. However, as the distance between sensor nodes increases, the minimum jump height required for successful communication becomes higher, (see Section 3.2.2). We call this minimum height, the threshold height. A jumping sensor needs to jump higher than the threshold height to provide enough time to communicate with others. The period of time the jumping sensor stays on and above the threshold height is called the communication time.

In order to calculate the required jump height of a jumping sensor such that it has enough communication time, let us study the jumping process from the moment the jumping sensor reaches the maximum height. At this point its velocity is zero and the sensor will begin to fall. The falling distance can be estimated using free fall Eqn. 3.4, where g is the gravity acceleration ($9.81 m/s^2$) and t is time in seconds while falling.

$$y = \frac{1}{2} g t^2 \tag{3.4}$$

The required jump height (H_{max}) is obtained by rewriting Eqn. 3.4 to consider the falling distance until reaching the threshold height (H_{Th}), Eqn. 3.5.

$$\begin{aligned} \Delta y = H_{max} - H_{Th} &= \frac{1}{2} g t_{\Delta y}^2 \\ H_{max} &= H_{Th} + \frac{1}{2} g t_{\Delta y}^2 \end{aligned} \tag{3.5}$$

Note that $t_{\Delta y}$ is the period of time it takes to fall from H_{max} to H_{Th}; by symmetry, the

period of time for the jumping sensor to reach H_{max} from H_{Th} is equal to the time it takes to fall from H_{max} to H_{Th}. Therefore, $t_{\Delta y}$ is half of the communication time.

Figure 3.4 is a plot of Δy, the height needed above threshold height. As an example, if 600 ms are required to communicate while airborne, a jumping sensor will need to jump 44.2 cm above the H_{Th} to guarantee the communication height.

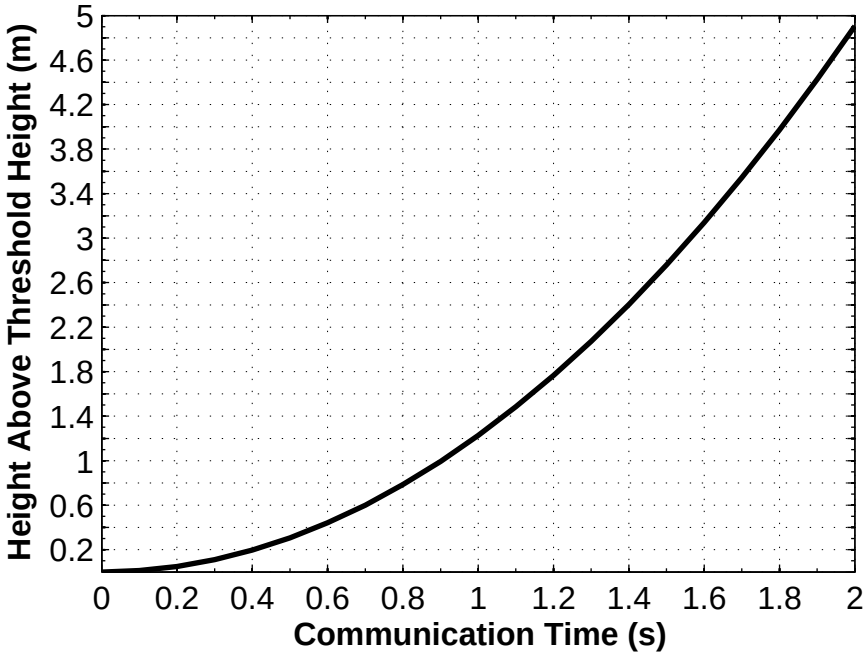


Figure 3.4: Height Needed Above Threshold Height VS. Communication Time

A jumping sensor may not have the control to jump to different heights, instead, it may be only able to jump to a fixed height. In such case, the available time will depend on the maximum height it can jump and the communication threshold height. Figure 3.5 is the plot of the available time for a jumping sensor that can jump to 1 meter height, given a threshold height. For example, if the minimal height for transmission is 30 cm, then the available transmission time will be the amount of time that the sensor takes to go up from this threshold height to one meter and come back to the threshold height, which is equal to 408.6 ms. Similarly, if the threshold height required to transmit is 50 cm, the available

transmission time would be 264.6 ms.

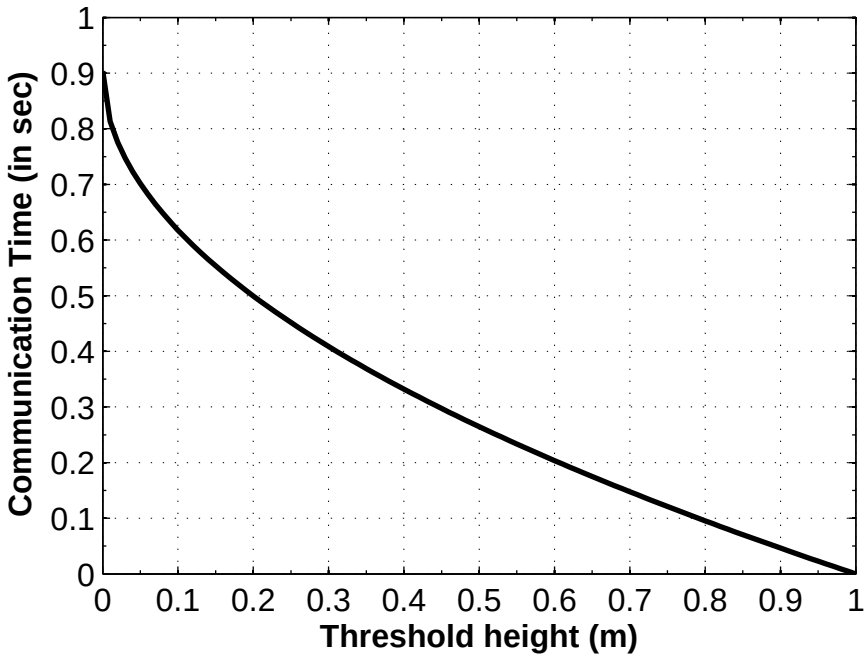


Figure 3.5: Time of Flight for 1 Meter Jump Height

3.3.2 Jumping Algorithm

With respect to the jumping characteristic of a jumping sensor, two cases are considered: the jumping sensor can either control the height it jumps or always jump to a fixed height; the last case being a specific case of the previous one. Having control of the jump height presents the opportunity to optimize the jump energy consumption.

The jumping sensor needs to jump high enough to stay above the communication threshold height for the required communication time, which depends on the transmission data rate of the sensor. When the required communication time is larger than the maximum time available for one jump, the data has to be partitioned in multiple jumps in order to be transmitted. Then, it is on the last jump that the height control capability will take place,

lowering the jump height when the remaining data to be transmitted requires a smaller communication time.

Let D be the amount of data to be transmitted, at a minimum threshold height h. Let D_h be the corresponding amount of data that can be transmitted on the available communication time with the highest jump height. The number of jumps is computed with Equation (3.6).

$$Jumps \ = \ \left\lceil \frac{D}{D_h} \right\rceil \tag{3.6}$$

When the jumping sensor can control the jump height, the height of the last jump can be adjusted depending on the size of the remaining data that has to be transmitted, Equation (3.7).

$$Last\ Jump\ Data \ \Leftarrow D \ mod \ D_h \tag{3.7}$$

If there is remaining data for the last jump, the new required communication time is computed, and the new jump height is obtained as presented in Section 3.3.1. It has to be noted that when the remaining amount of data for the last jump is zero, it means that all the jumps are performed with the highest jump height possible.

3.3.2.1 Jump Data Partitioning

The flow diagram for the data partitioning process is presented in Figure 3.6. The process begins with an amount of data that has to be transmitted. The feasibility box checks if all the data can be transmitted with the maximum jump height possible. If not, the data is partitioned. Each partition aggregates multiple packets, and the number of partitions is equivalent to the number of times the hopping sensor needs to jump.

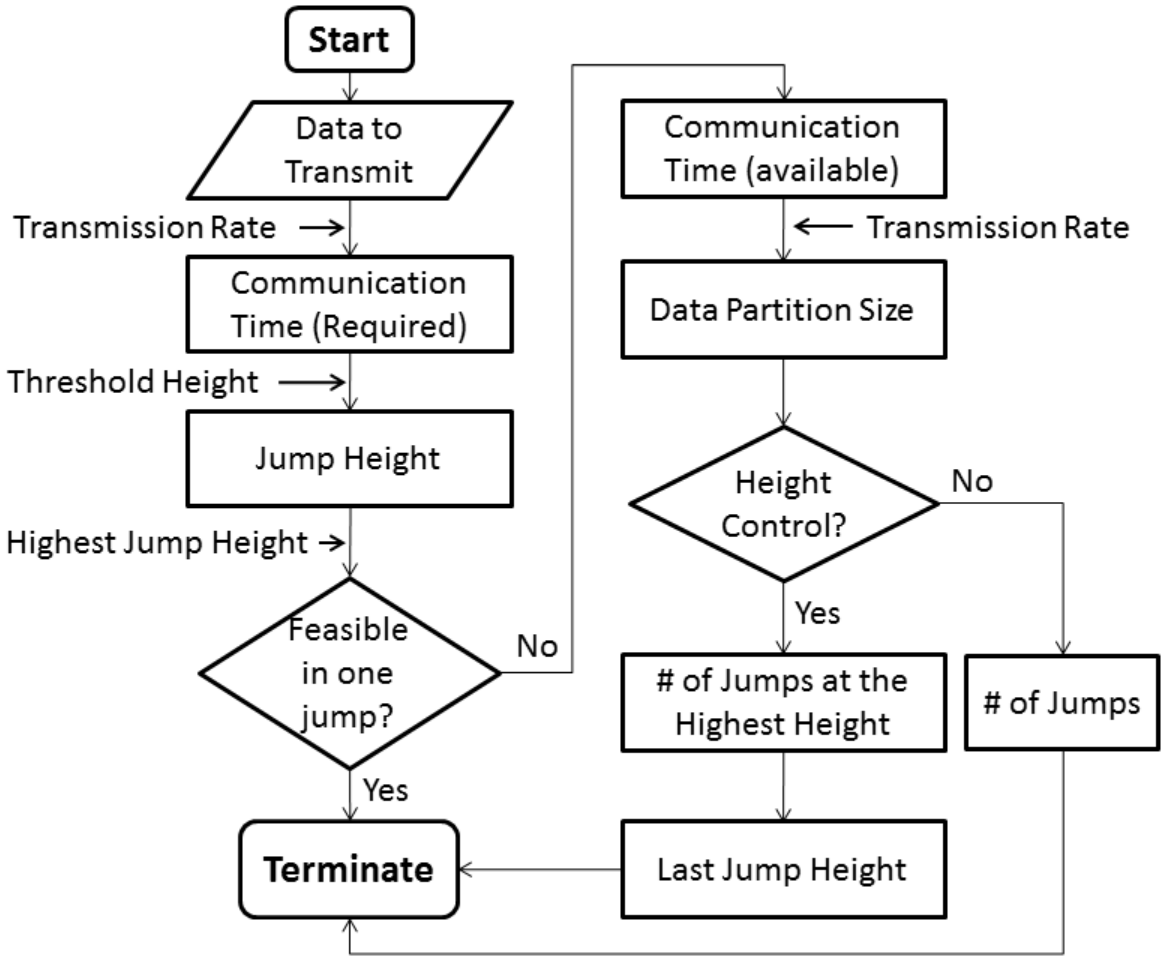


Figure 3.6: Jump Data Partitioning Flow Diagram

If the jumping sensor has jump height control, the number of jumps at highest height is provided, along with the last jump height. Otherwise, if the hopping sensor is limited to jump to a fixed height, a reduction in computation overhead will be seen by not requiring the extra calculation of the last jump height based on residual data.

3.3.3 Airborne Communication Process

The change in height from a jumping sensor creates a temporary increase on its wireless communication range. Next, an airborne communication process is presented to coordinate the jumping procedure with neighbor sensors, and to enable two-way communications in

jumping sensor nodes when airborne.

The airborne communication process is organized in two phases:

- Advertisement phase
- Data communication phase

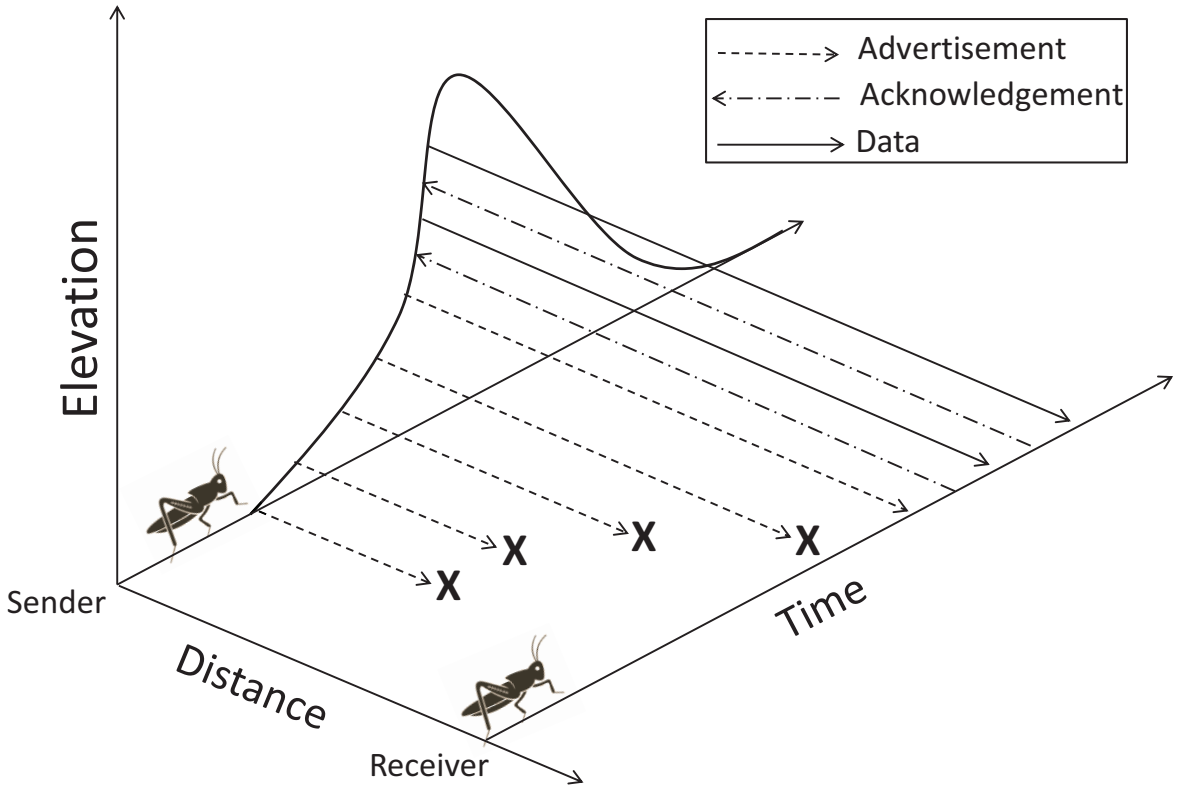


Figure 3.7: Airborne Communication Process

A jumping sensor starts the advertisement phase before it jumps, where it indicates a potential receiver of this action. The receiver in turn acknowledges the advertisement, indicating the hopping sensor to proceed with the transmission. To overcome the problem of out-of-communication-range-at-ground-level receiver, the advertisement is continued during the initial stage of the jump, until an acknowledgment is received. The advertisement phase is also used to synchronize the sender with the receiver when communication in multiple

jumps takes place. During data communication, if either a packet or an acknowledgment is lost, the transmitter will retransmit the packet after the expiration of a retransmission wait period.

The airborne communication process is depicted in Figure 3.7, where only the sender jumps. From the figure it can be appreciated that due to the long distance between the nodes, advertisement packets cannot reach the receiver. However, as the sender jumping sensor gets elevated, its transmission range increases allowing the advertisement packets to reach the receiver. The receiver acknowledges, and the data communication phase proceeds as described in Section 3.3.2.

3.4 Experimental Evaluation

In order to evaluate the airborne communication process, experiments were performed to collect measurements of signal strength, and register packet losses for packets transmitted while airborne.

IRIS Crossbow sensors were used, with a jumping emulator mechanism. The jumping action of a jumping robot was emulated using an elastic string mechanism, because of the unavailability of a jumping robot prototype able to meet desired jump heights. The mechanism was fixed to make the sensor to jump to a height of 1 meter with an error of ± 0.05 m (jump allowed between 0.95 m and 1.05 m). A total time of flight was registered to be on average 1 second. In most of the experiments, the jumping motion is brought to a stop on first ground landing, but in some circumstances the sensor had a bouncing effect resulting in a few additional milliseconds of elevated communication time.

3.4.1 Airborne RSS

Experiments were performed with different packet interval rates, 20 msec and 50 msec, to measure the changes in the received signal strength from packets transmitted airborne. The impact on RSS due to hopping was evaluated from 25 m onwards, to avoid possible communication at ground surface level. Multiple sets of experiments were performed and the results are presented in Figures 3.8, 3.9, and 3.10.

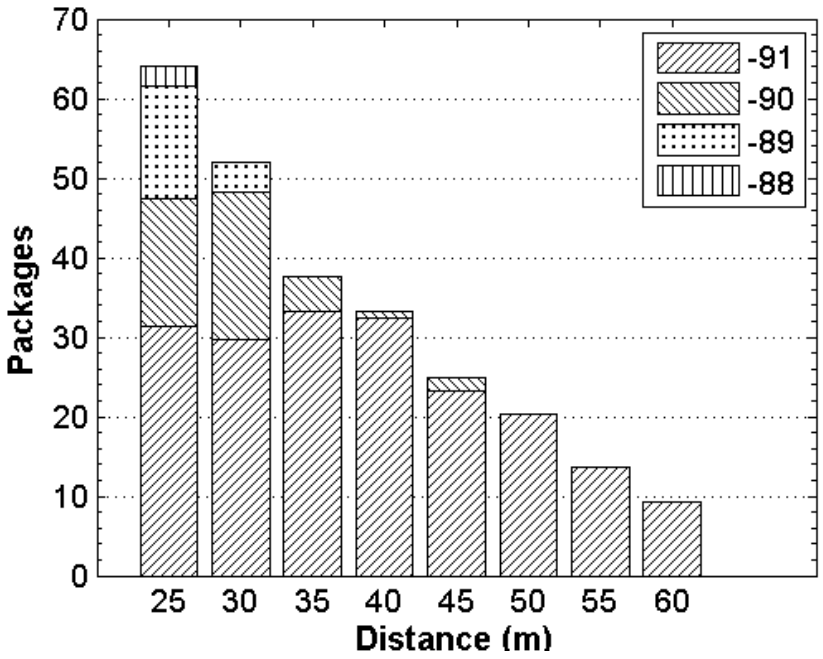


Figure 3.8: RSS Airborne Transmitted Packets (20msec rate)

In Figures 3.8 and 3.9, it can be noticed that during a single jump, the measured RSS varies between −91 dBm and −88 dBm, with the count of packets diminishing at higher dBm values. This is further demonstrated in the Figure 3.10, where the corresponding RSS of each packet transmitted airborne at a packet rate interval of 50 msec, for a distance of 25 m, is presented. The plot shows that with the practical increase in elevation, the received signal is stronger. Likewise, as the sensor starts descending, a reduction in signal strength

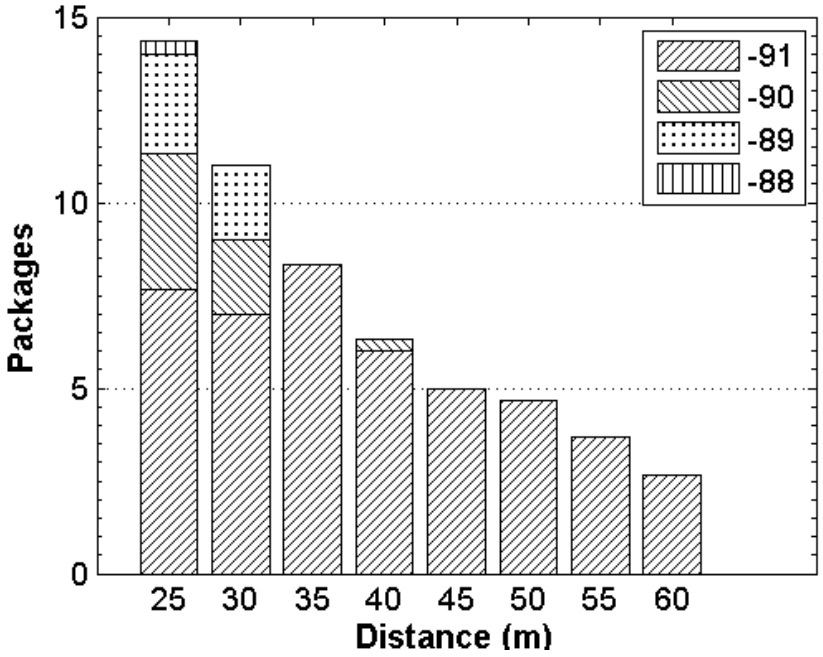


Figure 3.9: RSS Airborne Transmitted Packets (50msec rate)

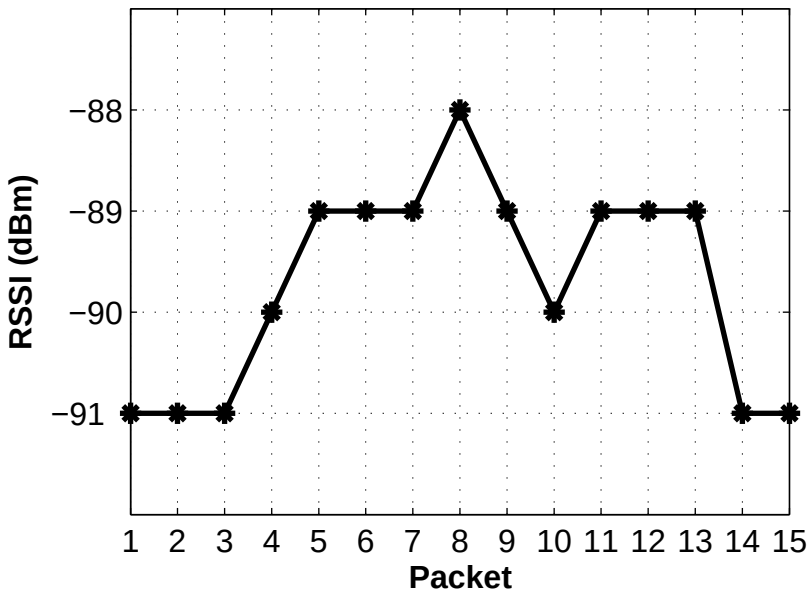


Figure 3.10: RSS Airborne Transmitted Packets at 25 meters (50msec rate)

is registered. The only blemish in the almost uniform plot is the RSS of packet 10, which for this experiment it is believed to be caused by the change in the sensor node orientation due to mid-air rotation while jumping.

3.4.2 Airborne Throughput

Airborne throughput was measured for packets transmitted while airborne during the data communication phase. For this experimental evaluation the sensor was set to transmit using two packet interval rates, 10 msec and 20 msec. The transmitter (while airborne) kept retransmitting the same packet until it received a valid acknowledgment of receival from the base station receiver fixed at ground level. On receiving a valid acknowledgment, the transmitter sent out the next packet in the queue.

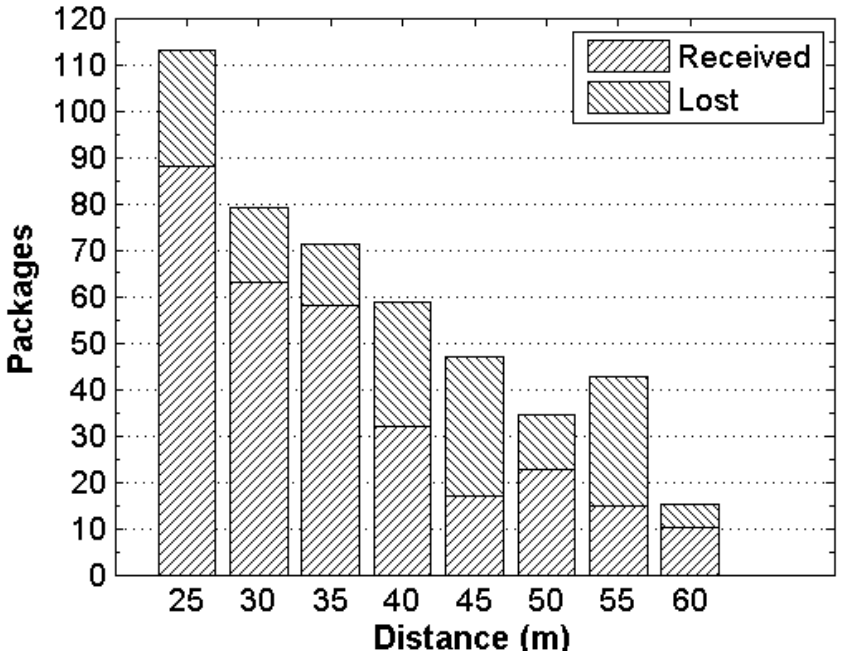


Figure 3.11: Throughput Airborne Data Communication (10msec)

Figure 3.11 shows the number of packets received and lost during the data communication phase for different distances between the jumping sensor and the receiver, for a 10 msec packet interval rate. As the distance increases, the total amount of transmitted packets (received + lost) reduces. This is as predicted, because the signal strength of the packets arriving at the receiver is reduced to a value below the sensing threshold for IRIS, making the advertisement

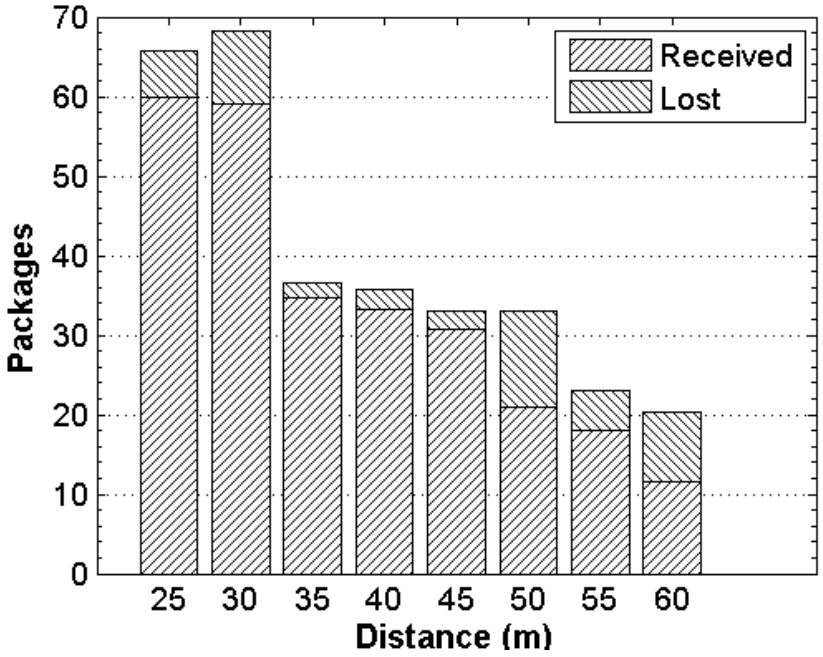


Figure 3.12: Throughput Airborne Data Communication (20ms)

phase to last longer, leaving less time for the data communication phase. The packet loss can not be fit to a predictable curve due to the rotation movement of the sensor in the air, which changes antenna orientation. Results from 20 msec experiments followed a similar trend, but with slightly less packet losses, Figure 3.12.

It is worth to note that the unevenness of the terrain, nearby structures and objects, and weather conditions have an effect on the signal propagation, and consequently, on the number of packets received.

3.5 Summary

In this chapter a comprehensive study of communication range improvement for a jumping sensor is presented. A discussion of the primary factors that impact the signal strength of wireless sensors is provided and studied through experimental evaluations. Field experiments were performed with wireless sensor nodes to measure the signal strength variance over distance for different terrains. The impact on communication while a jumping sensor jumps is presented. Results indicate that transmitting from higher than ground level increases the wireless communication range of the sensor. For a typical case, the communication range is increased from 5m-10m to 30m-60m, when the sensor jumps to an average height of 1 meter. Moreover, a two-way airborne communication process is defined and tested, proving to be effective as an alternative to reach distant receivers.

Chapter 4

HOPPING SENSOR NETWORK

4.1 Preliminaries and Motivations

Sensor nodes are in charge of monitoring their surrounding environment, resulting in generation of data. The gathered data will depend on the target application, e.g., environmental monitoring, event detection, etc. Furthermore, the data needs to be forwarded to a base station for data aggregation. Communication range is limited on sensor nodes, therefore, there is the need to have many deployed sensors in order aid in the routing of data to base station.

This chapter discusses the impact of network connectivity and area coverage in a jumping sensor network. A Hopping Sensor Network Model is defined to increase sensing area coverage, with the enhancement of network connectivity. A Hopping Sensor Routing Protocol is designed from the model, to balance the energy consumption on active hopping (jumping) sensor nodes. Both, the sensor network model and the routing protocol are evaluated through simulations. Discussion analyses for the obtained results are provided.

4.1.1 Chapter Organization

The relationship between wireless sensor network connectivity and effective area coverage is discussed in Section 4.2. The impact in network connectivity and area coverage from

employing jumping sensors in multiple network topologies is evaluated. The Hopping Sensor Network Model is defined in Section 4.3. The Hopping Sensor Routing Protocol is presented in Section 4.4. Performance evaluations of the routing protocol are provided. A chapter summary and conclusions are provided in Section 4.5.

4.2 Network Connectivity and Area Coverage

A single node is said to have connectivity if itself or within its neighbor nodes exists a direct or indirect communication path to the base station. Network connectivity is obtained with Equation (4.1), in where n is the total number of nodes in the network, and c is the number of nodes having connectivity considering our definition.

$$NetworkConnectivity\ \% = \frac{c}{n} * \ 100 \tag{4.1}$$

The area that has to be covered by a wireless network can be divided into unit regions. Each sensor node senses over a set of unit areas. For example if an area is composed of 10 unit areas, and each sensor node is able to sense over 2 unit areas, then theoretically, only 5 sensors are enough to sense the entire area. However, in practice, sensors cannot always be precisely positioned in needed locations, and not all of them may have connectivity. Moreover, considering that the resulting area covered is dependent to those sensors that can transmit data back to the base station, and the redundancy of sensor coverage of unit areas, the area covered does not increase linearly with the increase in connectivity. Nevertheless, increase of network connectivity has a significant impact in the total area covered.

4.2.1 Connectivity and Area Coverage Evaluation

Different scenarios were simulated to evaluate the impact of network connectivity and sense area covered by employing hopping sensor nodes. A field was simulated as a two dimensional grid space of 250 m x 250 m. This grid space is composed of small unit areas of 1 m^2. Sensors were randomly distributed over the field. The amount of sensors was determined based on the desired redundancy (R) of sensors covering a unit area, the grid space area (GA), and the sensing area of a sensor (SA) (eq. 4.2).

$$Number\ of\ nodes = \frac{GA}{SA} * R \tag{4.2}$$

A sensing range radius of 10 m was selected, yielding a rounded sensing area (SA) of 314 m^2. Sensor redundancies of 2, 3, 4, and 5 were used, obtaining 398, 597, 796, and 995 sensor nodes respectively. A base-station acting as a sink was positioned in the middle of one of the grid edges.

Network connectivity and area coverage was measured for twenty (20) network topological scenarios, for each sensor redundancy value. The case of sensors staying at the surface level (0 jump height) was considered as the base case for comparison. The succeeding scenarios are for nodes employing the jumping capability, each for a different maximum jump height. Only the areas covered by those sensors having connectivity are considered. Figures 4.1 and 4.2 present the network connectivity and area coverage percentages obtained, respectively.

Figure 4.1 shows the average of connectivity percentage results obtained in the simulation runs. It can be appreciated that for sensors communicating only at surface level, which means that their communication range is limited to 5 m (lowest packet losss range, see Section 3.2.2), the network connectivity on average is 0%. By increasing the jump height,

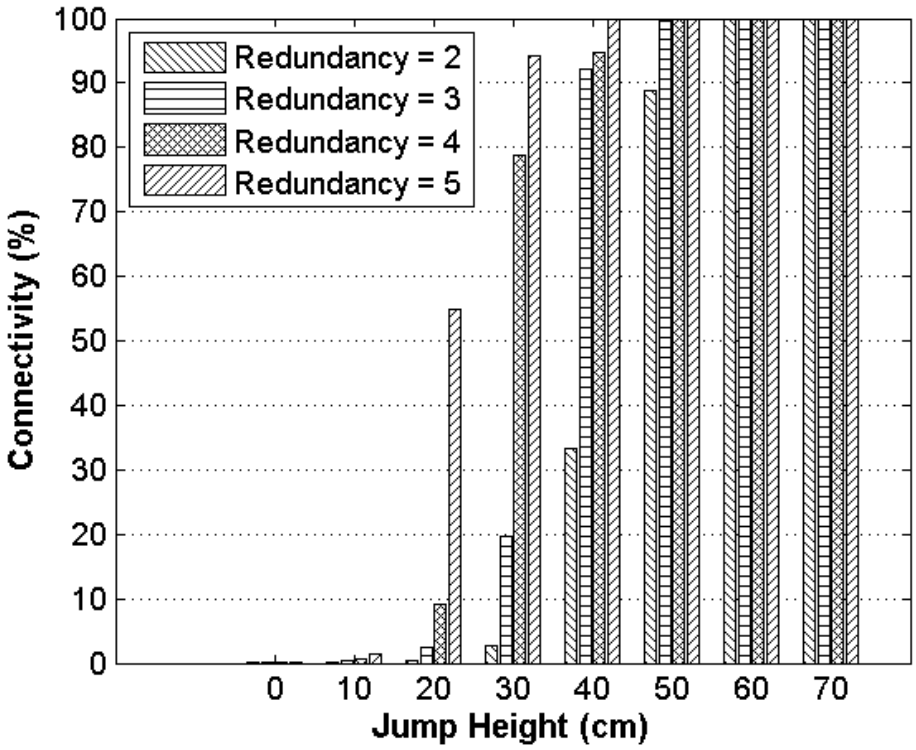


Figure 4.1: % Connectivity

the communication range of a sensor increases, resulting in an improved network connectivity. Another factor affecting the network connectivity is the redundancy coverage. Having high redundancy values will create high density of sensor per area, resulting in sensors to be closer to each other, leading to better network connectivity. Additionally, from the results it is seen that with a jump height of $40cm$ and higher, more than 90% connectivity was achieved for redundancies greater than 2.

The percentage of area covered results are shown in figure 4.2. The connectivity on the network has a critical impact on the area covered, since only those areas sensed by sensors having connectivity are considered covered. This effect can be seen on sensors communicating only at ground level, where the network connectivity was nearly 0%, resulting in a 0% average of area covered. A notable difference in area coverage is obtained when network

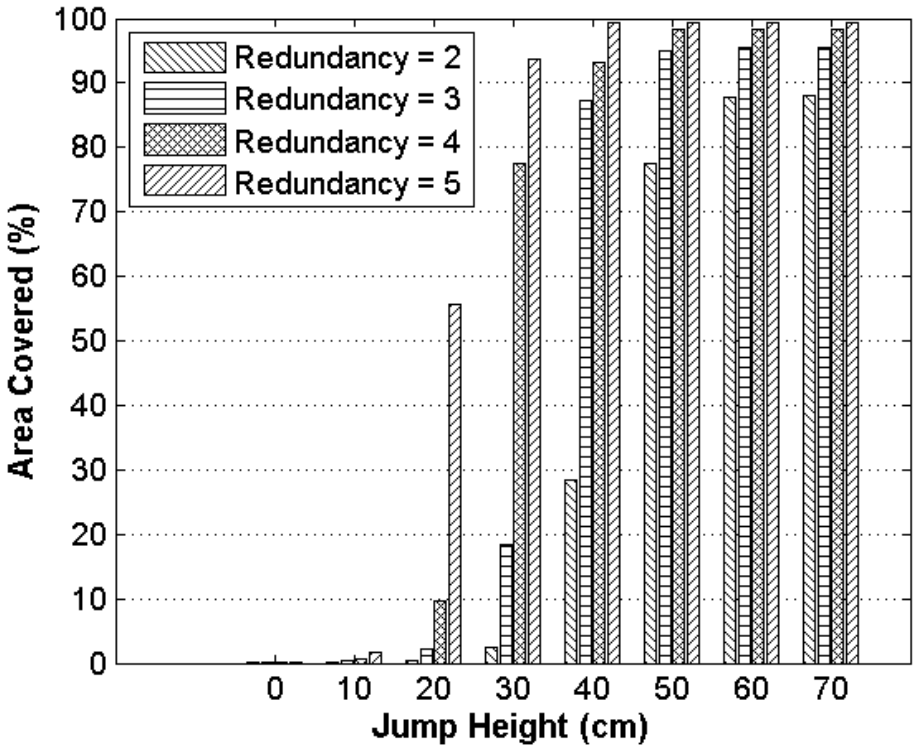


Figure 4.2: % Area Covered

connectivity is 90% (when the jump height is $40cm$) and above. This phenomenon can be better appreciated in Figures 4.3 (topology with $30cm$ jump height) and 4.4 (topology with $40cm$ jump height), where the total area covered is depicted (shaded area) for deployed topologies having a sensor redundancy value of 3.

The results indicate that a $40cm$ jump for this type of sensor node is a good threshold height to have sufficiently high connectivity and increased sensing area coverage. However, sensors may need to jump higher to obtain the desired amount of transmission time. Another factor of importance for the area coverage is the sensor topology. Since sensors were randomly positioned, sensor clusters were likely, i.e., areas with higher densities of sensors, leaving some other areas with fewer sensors.

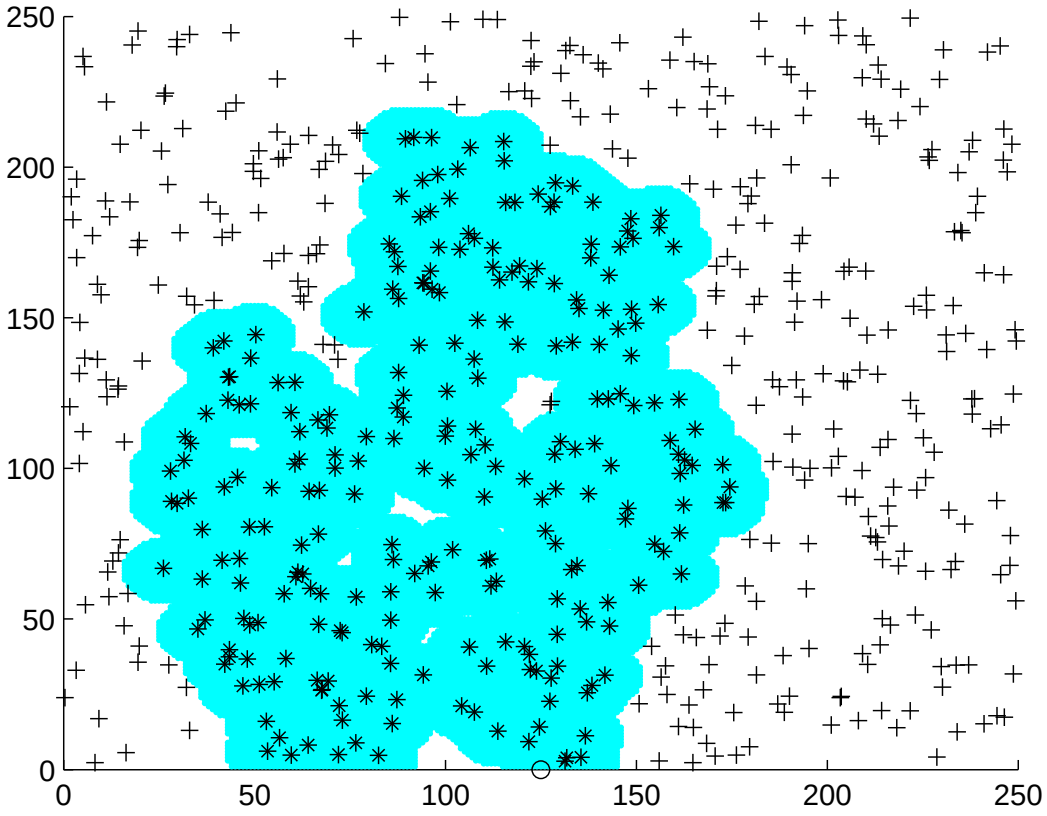


Figure 4.3: Topology Area Coverage for 30cm Jump Height

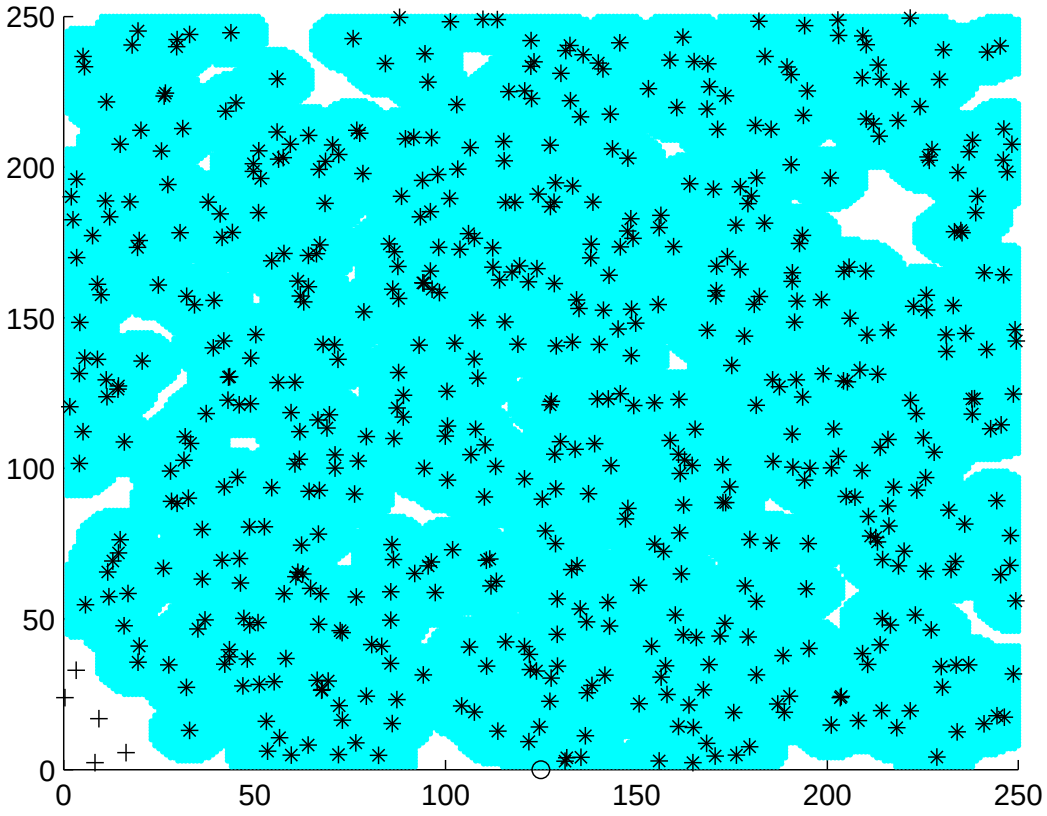


Figure 4.4: Topology Area Coverage for 40cm Jump Height

4.3 Hopping Sensor Network Model

In a mobile sensor network model, the premise for the sensor node mobility is to relocate the sensor node, thereby increasing the sensing coverage area. In contrast, a Hopping Sensor Network Model (HSNM) strives to increase the connectivity of the sensor nodes with the base station in order to increase the sensed area.

The design of a HSNM needs to take into consideration vertical and horizontal displacements, energy consumption, and different application requirements, Figure 4.5.

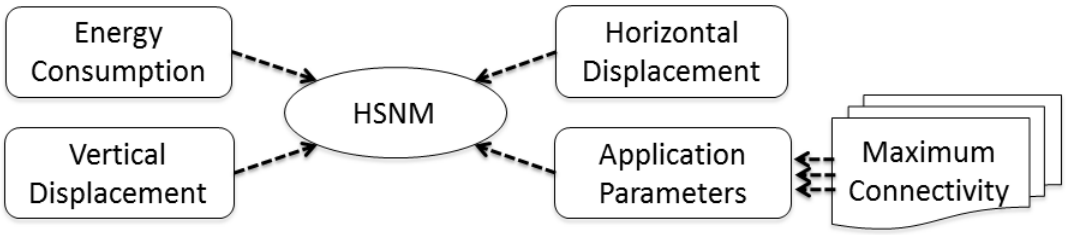


Figure 4.5: Hopping Sensor Network Model

4.3.1 Vertical Displacement

Vertical displacement is the height that a sensor node can reach hopping. This, along with the communication threshold height, influences the available time for transmission. With the knowledge of the available channel bandwidth, the total data that can be transmitted per hop is computed as in Section 3.3.

4.3.2 Horizontal Displacement

Horizontal displacement is the distance covered by each hop of the node and is difficult to be controlled. In the HSNM, the goal is to keep this parameter as close to zero as possible in

order to maintain the node position relative to its neighbors. In a flat terrain, if the number of hops is large, then the overall displacement will average out to zero due to random hop direction. But, this parameter gains importance in an uneven terrain.

4.3.3 Hop Energy Vs. Transmission Energy Consumption

A sensor node consumes energy to hop as well as to transmit data. The relationship between the two can be studied with a simple example. Let there be 10 nodes that are at a distance of 5 m from each other as shown in Figure 4.6. Results from practical experiments, presented on Chapter 3, showed the sensor communication range with lowest packet loss at different jumping heights. Let us consider the two heights, $0cm$ (no jump) and $60cm$, with communication range of $5m$ and $20m$, respectively.

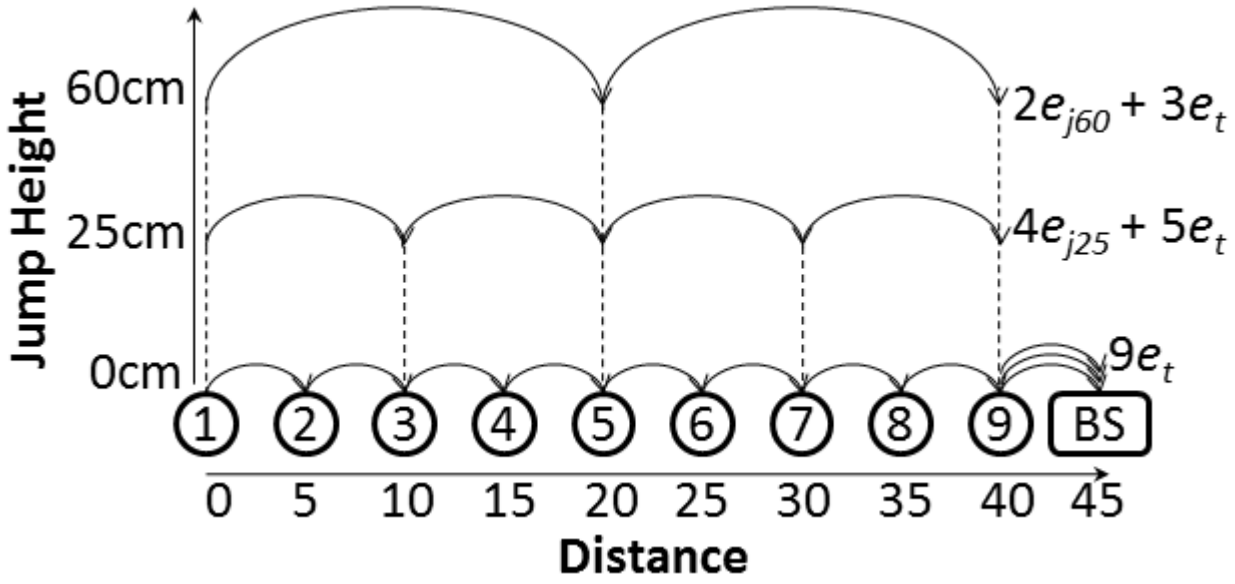


Figure 4.6: Hopping Sensor Energy Consumption

Let e_t be the energy consumed to transmit data over one hop and e_{j60} be the energy consumed to jump to height 60 cm. As depicted in figure 4.6, to communicate at the ground level consumes $9e_t$ units of energy. Consider the case when the nodes hop and transmit. Node

one (1) hops to height 60 cm and transmits the data directly to node 5 while consuming $e_{j60} + e_t$ energy. Similarly, node 5 communicates with node 9 by consuming the same $e_{j60} + e_t$ energy. Node 9 communicates with the base station consuming one e_t of energy. In this case, the total energy consumed is $2e_{j60} + 3e_t$. Similarly, if the node hops to height 25cm while expending e_{j25} energy and can reach a communication distance of 10m, then the total energy consumed for the communication will be $4e_{j25} + 5e_t$. Communicating while hopping can provide alternative routes for sensors to communicate with the base station, while avoiding the overload of nodes in high traffic routes.

4.3.4 Application Parameter

The application parameters are the variables that are regulated by the application's needs. In a WSN there are two basic parameters of usual interest, namely coverage and connectivity. Certain non-critical applications such as a temperature sensing in an open environmental field will tend to require a maximum coverage while requiring an intermittent connectivity. This is sufficient because the sensor nodes can sense over longer periods of time and then transmit their data less frequently. In contrast, a real time application (e.g., surveillance monitoring) requires a maximum coverage while having a continuous connectivity. Application parameters are on of the main factors that influence the impact on performance and lifetime of the network.

4.4 Hopping Sensor Routing Protocol

A Hopping Sensor Routing Protocol (HSRP) is a centralize approach based on the HSNM. The base station decides the routing path of the packet within a time frame, and provides the

best energy preserving path for the packets. The decision making process involves multiple stages and is depicted in Algorithm 4.1. Let P_m be the paths available from a node m to the base station, with each path P_i consisting of a set of nodes with their respective required jump heights for the given path. A node can participate in more than one path. The remaining energy for a node k is expressed as e_k, the energy consumed to jump to height h is e_{jh}, and e_t is the energy for transmitting.

The HSRP finds the minimum energy consuming path while not depleting any node beyond a threshold τ of energy consumption. This threshold is the maximum desired operational energy cost (fraction of energy required out of their remaining energy) per node in a path. This value is determined based on application needs and is set prior to execution. The permissible amount of energy consumption of a node in a path is dependent on the residual energy in the node. With aging of network life, the remaining energy of the nodes decreases and will result in some nodes not meeting τ. In order to minimize the impact of this, the algorithm, as last resource, selects the best among the high energy consumption paths.

Algorithm 4.1 Hopping Sensor Routing Protocol

Input: $Paths_i(n),\ e_k,\ \tau - Energy\ threshold$
Output: Path(): selected path for all n nodes

```
m = 1
while m <= n do
    MaxEPathFrac = 1

    P_m = Set of paths for node m

    for all path P_i ∈ P_m do
        for all node k ∈ P_i do
            E_i = E_i + e_jh + e_t

            EnergyFraction(k) = (e_jh + e_t) / e_k

            if EnergyFraction(k) > τ then
                PathValidity(P_i) = False

                if EnergyFraction(k) > MaxEFrac_i then
                    MaxEFrac_i = EnergyFraction(k)

                end if
            end if
        end for
    end for
    SortedP_m = sort(Paths P_m on Energy E)

    while (!PathFound) do
        P_i = SortedP_m{next path}

        if PathValidity(P_i) = True then
            PathFound = True

            Path_m = P_i

        else if MaxEFrac_i < MaxEPathFrac then
            Path_m = P_i

            MaxEPathFrac = MaxEFrac_i

        end if
    end while
    Path(m) = Path_m
end while
```

4.4.1 HSRP Evaluation

The HSRP was simulated to evaluates its performance on network energy consumption. The values obtained from the practical in-field experiments were used to determine the communication ranges of the nodes. The base case for the communication was when the sensor node jumped to 30 cm height. At this height, the IRIS Crossbow sensor is able to communicate over a distance of 12 m as compared to ground level, where it could only communicate up to 5 m without packet losses. At 5 m communication range most of the nodes are completely disconnected, hence prompting us to use a sensor jump height of 30 cm as the default base case. This was compared against the height when the sensor node can jump either 30 cm or 60 cm. The decision of whether the sensor should jump to 30 cm or 60 cm was taken using HSRP. When the node jumps to 60 cm height, it is simulated to consume double the amount of energy compared to when it jumps to 30 cm. Irrespective of whether the node jumps to 30 cm or 60 cm, the transmission cost, in terms of energy, remains the same as the node always transmits at the highest power (3 dBm). The debate between the transmission cost and the hopping cost is out of the scope of this evaluation, since without the hop, the resultant network is mostly disconnected and the sensed data cannot be transmitted back to the base station.

Every node started with 1000 units of energy. The energy threshold value for the node was a varying factor and was set to 0.005, 0.01, 0.02, 0.03, 0.04, 0.05. The simulations were run over 100 time periods, and during each time period 20% of the nodes were randomly selected to communicate.

Geographical routing protocol was used as base case to compare the energy consumption performance of HSRP. It uses the shortest distance as the criterion metric to select the node's

communication path to reach the base station. The energy consumption of the network was evaluated over the complete time length of simulation, Figure 4.7.

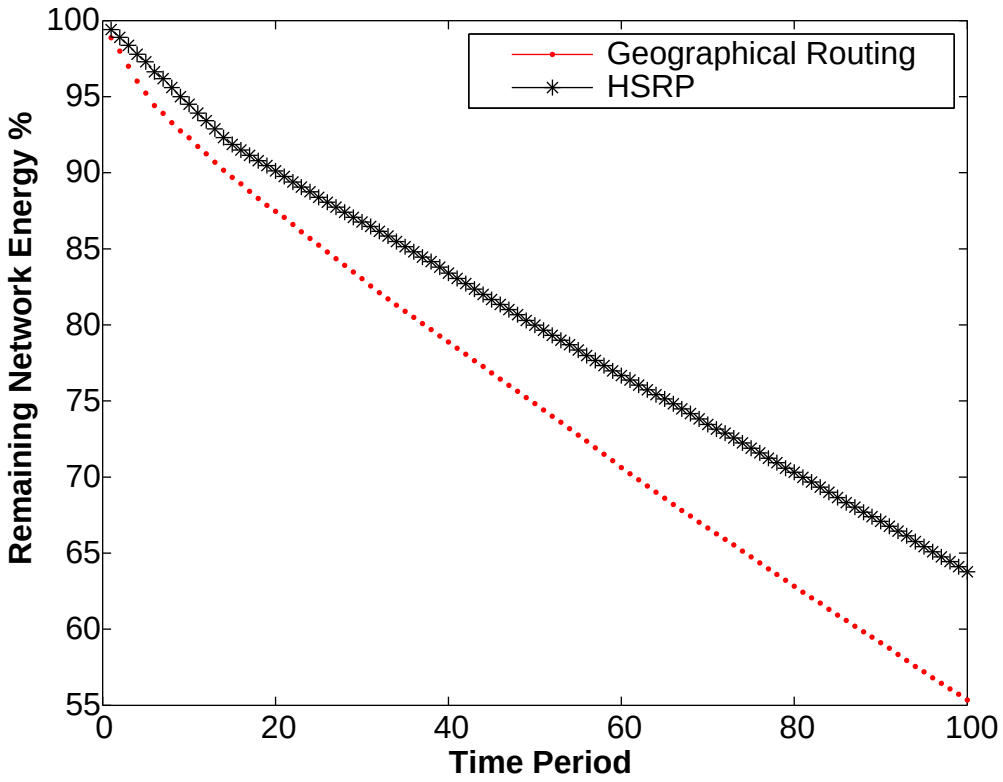


Figure 4.7: Network Energy Consumption

The energy consumed by the network employing HSRP is 20% lower compared to using geographical routing. This is because HSRP calculates the improvement of path quality by the nodes jumping to a higher elevation and selects paths with better quality. The quality is defined as the amount of energy savings that the path provides to the network by not depleting any node on the path with a significant portion of its remaining energy.

The second feature of critical importance is the energy balancing within the network. The number of dead nodes was studied over the course of the 100 time periods. We observed that the protocol balances the energy used in the network by the different sensor nodes. A node with a lower amount of energy is used less frequently to forward a packet. This significantly

affects the nodes closer to the base station. For example: under certain circumstances, by employing the protocol, nodes that are two hops away from the base station can hop up to $60cm$ height and transmit packets directly to the base station, thereby preventing the complete drainage of energy from nodes closer to the base station. The number of dead nodes in the case of HSRP is significantly lower, 1/8 of when using geographical routing. Also, without the use of the protocol, a large number of nodes closer to the base station consumed all their energy by forwarding packets for other nodes that were farther away, resulting in the reduction of connectivity, which affects the number of packets delivered.

4.5 Summary

In this chapter the jumping sensor network connectivity and area coverage are evaluated. A Hopping Sensor Network Model (HSNM) has been introduced in order to increase the connectivity of the sensor nodes with the base station, and thus, to increase the network sensed area. Furthermore, a Hopping Sensor Routing Protocol (HSRP) is designed from the HSNM with the main purpose of manage the energy consumption in the network. In fact, the protocol chooses the best path in terms of energy consumption, avoiding the fast energy depletion of active jumping sensor nodes. Simulation results showed that HSRP has lower energy consumption rate than geographical routing protocol, giving a longer network utile life.

Chapter 5

WIRELESS SENSOR NETWORK CONNECTIVITY INCREASE WITH JUMPING SENSORS

5.1 Preliminaries and Motivations

Network connectivity in a Wireless Sensor Network (WSN) is crucial for network operation. When wireless sensor nodes are randomly deployed in a field, there may be areas that lack sensor nodes. These areas tend to create *communication holes*, and in the worst case, break the topology into disconnected clusters of nodes. Figure 5.1 depicts a WSN topology of $200m \ x \ 200m$ area with a single base station located at the middle of the bottom edge; nodes with connectivity are marked with diamond shapes, and nodes without connectivity are marked with asterisks. All the nodes are transmitting at maximum power, however due to communication holes, only 31% network connectivity is achieved. From the figure, the formation of clusters of nodes can be identified as those disconnected groups (from base station) of interconnected nodes.

Jumping sensors can be used to increase connectivity in wireless sensor networks. However, a topology of jumping sensor nodes configured to enhance network connectivity by

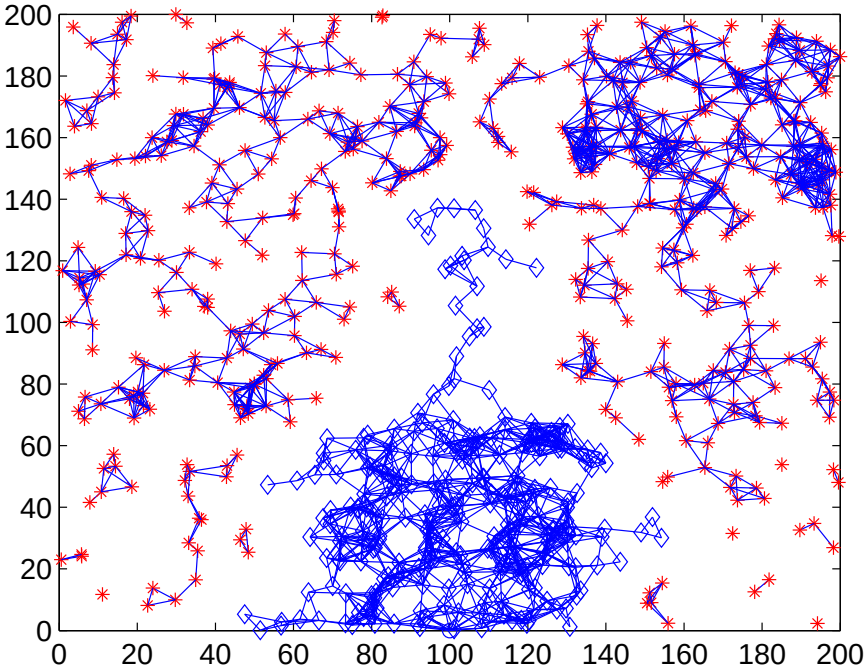


Figure 5.1: WSN Topology at ground level (31% connectivity)

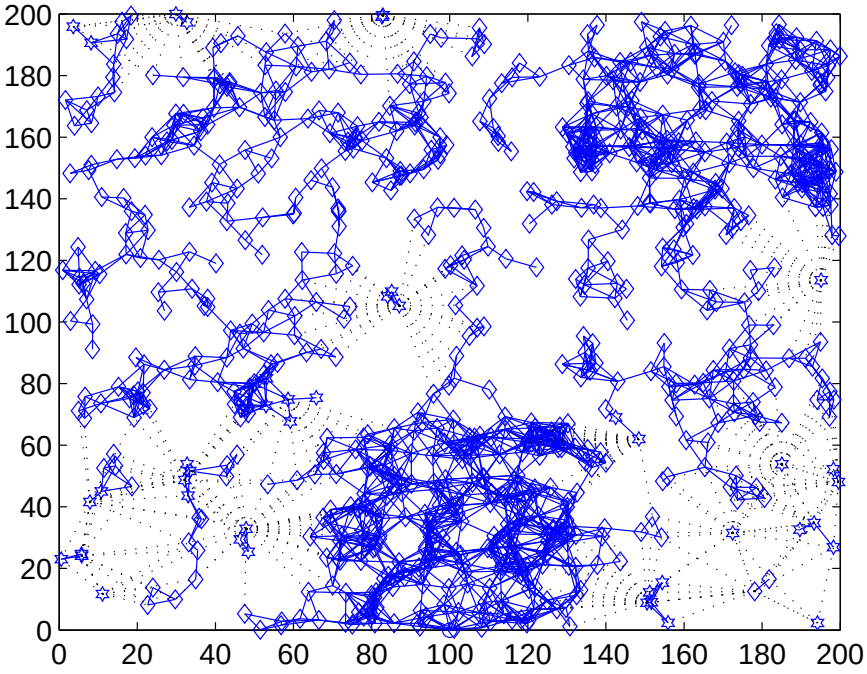


Figure 5.2: WSN Topology with gateway jumping sensors (100% connectivity)

means of jumping can significantly reduce the network life span, if not well managed. This chapter presents an approach to deal with communication holes, to enhance the network connectivity using selected jumping sensors nodes located at the boundaries of clusters of nodes. With the help of selected jumping sensors serving as gateways on disconnected clusters, 100% of network connectivity is achieved in the same topology (Figure 5.1). This improvement on network connectivity is depicted in Figure 5.2, where wireless links established by jumping sensors are represented as dotted lines.

In this chapter, an overview to the jumping sensor network topology is given. A survey on boundary node identification algorithms is provided. Two decentralized algorithms that use the jumping capabilities of sensor nodes are designed for the discovery of neighbor clusters of sensors, Boundary Node Aggressive Neighbor Discovery (*aggressive*) and Boundary Node Smart Discovery (*smart*). Furthermore, energy balanced packet forwarding schemes with the use of jumping sensor gateways are evaluated for cluster-to-cluster (C-to-C) communication.

5.1.1 Chapter Organization

This chapter is organized as follows. Jumping sensor topology is presented in Section 5.2. Boundary nodes identification algorithms are reviewed in Section 5.3. Node discovery algorithms are defined in Section 5.4. Evaluation of the algorithms and analysis of network performance employing C-to-C communications are presented in Section 5.5. A chapter summary and conclusions are provided in Section 5.6.

5.2 Topology

The deployment phase and characteristics of the geographic area impacts a wireless sensor network topology greatly. A common method of sensor deployment is to locate sensors precisely in the field to guarantee sensor connectivity with a base station, while maximizing the covered area. This method becomes infeasible when the network scales to cover difficult-to-reach or hazardous areas, e.g., volcano monitoring discussed by Song *et al.* (2009). As an alternative, sensor motes are deployed from the air; a technique sometimes referred as air-dropped-sensors, where sensors are carried and dropped by a helicopter, or an airplane to the target area. This technique gives the opportunity to reach those difficult-to-reach or hazardous areas. However, there is chance for sensors to land into segregated groups of sensors that may not have a connectivity path to the base station.

In this work, a **cluster** of sensor nodes is defined as a group of interconnected nodes relying on their wireless communication range at ground surface level, and packet routing capabilities. A cluster can have *direct connectivity*, i.e., one or more nodes from the cluster can reach directly the base station, and provide connectivity to others within the cluster. Similarly, the connectivity propagation concept applies between clusters. A cluster with *direct connectivity* will provide connectivity to its neighbor clusters, and those will continue in turn to form a cluster-to-cluster (C-to-C) path to base station.

With the existence of clusters in the topology, sensors nodes that lay on the edge (frontier) of a cluster of nodes are classified as **boundary nodes**. These nodes will help in the discovery of neighbor clusters and form **bridge** nodes, i.e., those boundary nodes that can successfully communicate with nodes from other clusters, by employing jumping sensor airborne communication. There may be more than one bridge node that shares the same

intended cluster receiver. Moreover, a cluster may be able to reach base station trough different clusters. In order to maximize network functional lifetime, a cluster can distribute the packet forwarding task among bridge nodes, by selecting a subset of bridge nodes as **gateways** to perform C-to-C packet forwarding.

5.3 Boundary Nodes

Cluster's boundary nodes provide the highest successful discovery probability because of their position relative to others. Therefore, selecting only boundary nodes to perform discovery, increases the probability of discovery, and reduces the network energy cost. The identification of boundary nodes has been addressed by Sahoo *et al.* (2007), and Fayed and Mouftah (2009).

Two kinds of boundary detection algorithms were introduced by Sahoo *et al.* (2007): sequential (SBNS) and distributed (DBNS). Furthermore, using these boundary nodes, a target detection protocol is used to know the entry and the exit of single target through the monitoring region. In selecting the boundary nodes, the two algorithms have similar performance. The main differences between them is the lower communication cost of DBNS compared to SBNS due to the less amount of information exchange and the faster selection procedure in DBNS regardless the increase of network size.

Fayed and Mouftah (2009) proposed a local convex view (lcv) as a means to identify nodes close to the boundary. It is a localized algorithm and can establish neighborhood coordinates if location information is not prior available. To check if a node is on the boundary, a node is verified to lie on the convex hull built with its one-hop neighborhood. The most important result is that lcv seems unaffected by position estimation error because of its geometric properties. However, it has difficulties identifying boundary nodes under special scenarios

where there are areas without sensors (*communication holes*) within the topology.

Boundary nodes have shown to be beneficial in wireless network topologies. They have been used to help other nodes in the localization process on wireless sensor networks, Du *et al.* (2007). Another approach presented by Aissani *et al.* (2008), employs boundary nodes to avoid routing packets to dead-end routes. Han and Poor (2009) introduced a method based on coalition games, in where boundary nodes are used to cooperate with nodes in the middle of the network to enhance network performance and connectivity.

5.4 Discovery

When a jumping node jumps, it has the chance to reach farther nodes, by employing airborne communication (see Section 3.3). Following this approach, jumping sensors can be used to discover and establish communication with other sensors that are far and without base station connectivity.

5.4.1 Boundary Node Aggressive Neighbor Discovery

Boundary Node Aggressive Neighbor Discovery is a decentralized algorithm that utilizes all the jumping sensor boundary nodes (BNs) from clusters, to jump and discover surrounding neighbor nodes belonging to other clusters.

In order to propagate the connectivity on the wireless network, this process begins on the cluster with *direct connectivity*, and continues on other clusters as they become discovered, in a sequential manner.

Every BN within a cluster will perform the cluster discovery, which consists of hopping (jumping), transmitting a discovery message, and listening for any reply while airborne. In

order for a BN to perform cluster discovery, it has to follow the following phases: ***Pre-hop, Hopping, Post-hopping***.

5.4.1.1 Pre-hop

Before a BN jumps, it has to coordinate with its one-hop[1] away BN neighbors, to avoid these BNs to jump at the same time and interfere with each other.

Every BN waits for a random amount of time before trying to jump and discover. Once this time is expired, the BN sends a *Hop_Advertise* packet to its one-hop BN neighbors, including its NID. Those BNs receiving this packet, reply with a *Hop_Ack* packet, including their node identification (NID) and the NID of the BN initiating the discovery. These BNs constrain themselves to perform hopping discovery while the current discovery is taking place. Upon receive of these *Hop_Ack* packets, the BN discovery initiator verifies if all its BN one-hop neighbors replied and proceeds to hop.

One or more BN neighbors may fail to generate the *Hop_Ack* packet, due to a packet loss or due to competition for hopping at the same time. To address the case of missing acknowledgement due to packet loss, the BN initiator waits a random amount of time and resends a *Hop_Advertise* packet to those missed BN neighbors, who in turn will reply with a *Hop_Ack* packet.

A similar procedure applies if the BNs were competing for discovery. Each competing BN is set to wait before attempting to re-advertise, to provide a chance for the other competitor to generate the *Hop_Advertise* packet first. Upon receipt of the advertisement packet, any competitor yields and replies with a *Hop_Ack* packet. The winning BN proceeds to hop, and those BN neighbors that exclusively received the advertisement packet from BN competitors

[1]One-hop communication refers to direct communication between two nodes.

and not from the winning BN, will reset their wait timers to start competing among them.

5.4.1.2 Hopping

While the BN is airborne, it transmits a *Discovery_Msg* packet, including its CID and connectivity information, i.e., *direct connectivity*, or the number of intermediary clusters to reach the base station. Any BN belonging to another cluster that is able to receive this packet will attempt to reply with a *Discovery_Reply* packet, including its CID and connectivity information.

5.4.1.3 Post-hopping

BNs on each cluster with discovery information become bridge nodes, and advertise it to the BNs of the cluster with a *Bridge_Advertise* packet, including its NID, node remaining energy, neighbor CID, and connectivity information of the neighbor cluster. In addition, clusters that are discovered will start to perform discovery, to propagate connectivity.

After the current discovery process finishes on a BN, the BN neighbors resume their wait timers for discovery. Those BNs that performed the discovery will not compete for discovery until a new discovery process is triggered, i.e., in the event a cluster gets divided due to communication holes resulting from missing nodes. It is worth noting that after a jumping node jumps, it may be possible (depending on the geographic area) to land in a position within the communication range of the two or more clusters at surface level; in such case, clusters are merged into one.

5.4.2 Boundary Node Smart Neighbor Discovery

In order to reduce the energy consumption of the discovery process, the discovery algorithm is enhanced to find surrounding neighbor clusters with a subset selection of BNs.

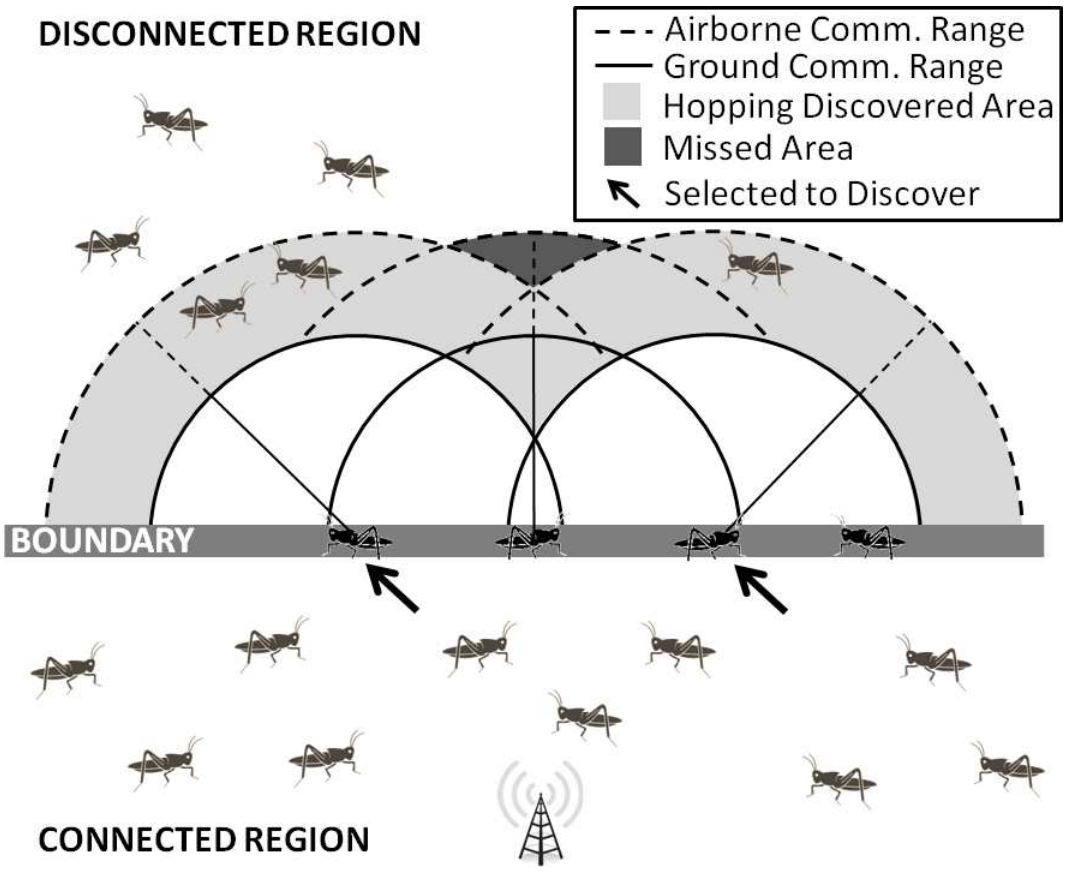


Figure 5.3: Discovered Area Gain of BNs Subset Selection for Discovery

The approach relies on the fact that once a BN is selected to perform discovery, its one-hop BN neighbors will not contribute much to expand the discovered area. Figure 5.3 depicts the gain in discovered area by only selecting BN that are not one-hop neighbors of the BNs already selected to perform jump discovery. The communication range of BN jumping sensors at ground level is represented by a solid line circle, while the communication range when hopping (jumping) is represented with a dotted line circle.[2] From Figure 5.3 it is also depicted the resulting missed area by not selecting the one-hop BN between the selected

BNs. However, the contribution to the communication area of the missed area, in most of the cases, is minimal.

In order to provide a BN subset selection, the process begins from the cluster having *direct connectivity*, where all BNs compete to perform discovery, similarly to the Aggressive Neighbor Discovery, discussed in Section 5.4.1. The main differences are on the BN behavior during the *Pre-hopping* and *Post-hopping* phases.

In the ***Pre-hopping*** phase, once a BN sends a *Hop_Advertise* packet, all its one-hop BNs will not compete for further discovery until another discovery event is required. BNs that did not receive an advertisement packet will continue competing for discovery by repeating the process. In this manner, a subset of BNs is selected to perform the discovery.

During the ***Post-hopping*** phase, if the discoverer BN does not reach any other cluster, it sends a *Discovery_Null* packet to its one-hop BN neighbors, and stops competing to do further discoveries until a new discovery event is triggered. Upon receipt of this packet, one-hop BN neighbors will reset their wait timer to compete for discovery. On the other hand, if a cluster is discovered, the BN advertises the discovery to all the BNs with the *Bridge_Advertise* packet. Those one-hop BN neighbors of the discoverer BN will stop competing for discovery. For the discovered cluster, the set of BNs able to successfully receive the *Discovery_Msg* packet become bridge nodes. These nodes advertise their new status to all the BNs with the *Bridge_Advertise* packet. BNs that are a one-hop neighbors of BNs that became bridge nodes will refrain from discovery until a new discovery process is triggered.

[2]Figure 5.3 is meant for visualization purpose only and is not drawn to scale. Jumping communication range has been measured to increase by a factor of 6 when hopping (see Chapter 3).

5.5 Simulation Results and Analysis

Simulations were performed to test the discovery algorithms in terms of energy consumption and connectivity propagation for a jumping sensor network. The life span of the network was tested with different configurations of C-to-C packet forwarding. Experimental data results were used as input for the communication ranges, 10 meters ground level communication, and 30 meters for airborne communication (see Section 3.4). The airborne communication range limit of 30 meters was selected as a conservative measure to account other factors, such as terrain and weather conditions, that can affect wireless communication.

5.5.1 Experimental Setup

Topology: Twenty (20) sensor network topologies were generated representing a geographic area of $200m$ x $200m$. Nodes were positioned following a random uniform distribution. Each node was simulated to provide sensing coverage to a circular area with a 7 m radius. The number of nodes in the network was selected to provide a sensing coverage of 3 nodes per unit area (1 m^2), resulting in 781 nodes (all jumping sensors). A base station node was located in the middle of the bottom edge of the square topology acting as a sink.

Energy: Each node was simulated to have the same amount of energy as the jumping sensor prototype built by Zhao *et al.* (2009). It uses a 3.7 V, 100 mAh LiPo battery, for a total stored energy of 1332 J. Their prototype can make jumps up to a height of 0.353 m, consuming 882.6 mJ of energy for each jump. A similar design of a jumping sensor able to jump 1 m height, is estimated to consume 1.775 J of energy. For simulation purposes, we considered an energy consumption of 2 J for the jump operation. Other node's energy consumption operations were determined following IRIS data sheet specifications as follow:

energy to transmit a packet of 36 bytes at maximum transmission power (3 dBm) is 0.09 mJ, and the energy to receive a packet is rounded to be the same as to transmit, since the different is relative small.

5.5.2 Discovery Algorithm Performance

In order to study the discovery algorithms performance, energy consumption and connectivity propagation were analyzed on the 20 network topologies after initial discovery. The algorithms were compared with all jumping sensors randomly employing individual discovery (referred as ALL). Table 5.1 presents the mean connectivity of network after the discovery phase, and the mean network energy cost in Joules for each algorithm. It can be appreciated that as the discovery algorithms become smarter with the selection of nodes to perform discovery, the energy consumption reduces without compromising connectivity.

Table 5.1: Discovery Algorithm Performance

Algorithm	Connectivity %	Energy Consumption (J)
ALL	100	1562.0
AGGRESSIVE	100	670.5
SMART	100	330.6

5.5.3 Network Performance

The network performance was analyzed over 5000 time period events for each of the network topologies. For every time period, a percentage of the nodes with connectivity was randomly selected to generate packets. Simulations were run separately for five different percentages of nodes generating packets (communication loads): 20%, 10%, 5%, 2%, and 1%.

Boundary C-to-C packet forwarding was employed with jumping sensor acting as gateway nodes to reach the base station. A delay transmission protocol was employed for jumping

sensors airborne transmission. Packets arriving at gateways (nodes that need to jump and transmit) were buffered until transmission buffer became full, but not for more than 3 time periods after arrival to the node, then transmitted airborne.

Two configurations were simulated to evaluate the network performance employing C-to-C packet forwarding: *single* gateway per cluster, and *multiple* gateways per clusters. They were compared with packet forwarding without gateways (no-cluster organization). *Single* gateway per clusters is an energy preserving approach, in where every discovered cluster elects a single gateway node to forward packets to neighbor clusters, to reach the base station. Shortest path and remaining node energy were used as the gateway selection criteria, smallest NID was used to break any tie. *Multiple* gateways per cluster is an extension of the *single* gateway approach, but less energy conservative. It provides alternative routes to nodes in a cluster to obtain shorter paths length to reach base station trough different gateways. With packet forwarding without gateways, every jumping sensor node takes advantage of reaching the farthest nodes with airborne communication to attempt shortest paths routes to base station. It is worth noting that if a jumping sensor is not able to shorten the path with airborne communication, then it employs ground level communication.

Each C-to-C packet forwarding configuration was evaluated with three metrics:

1. ***Remaining Network Energy***, mean of the remaining energies in all the nodes, at the end of simulation, with respect to the total initial network energy.

2. ***Remaining Network Connectivity***, percentage of nodes left with connectivity to base station.

3. ***Mean Packet Delay Time***, mean of the time periods that each packet required to arrive at base station.

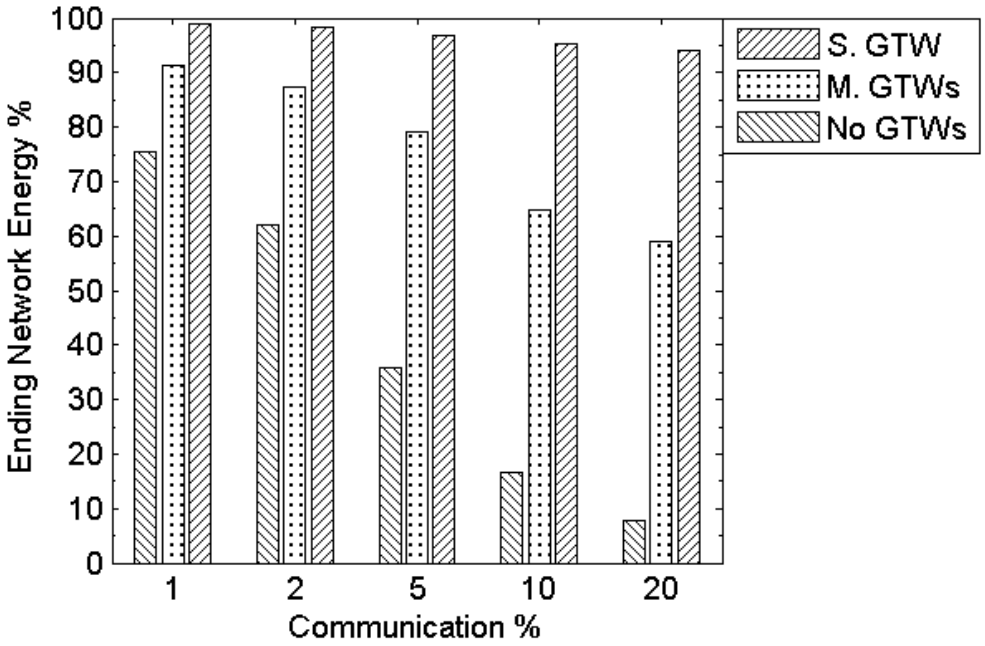


Figure 5.4: Remaining Network Energy for Different Communication Loads

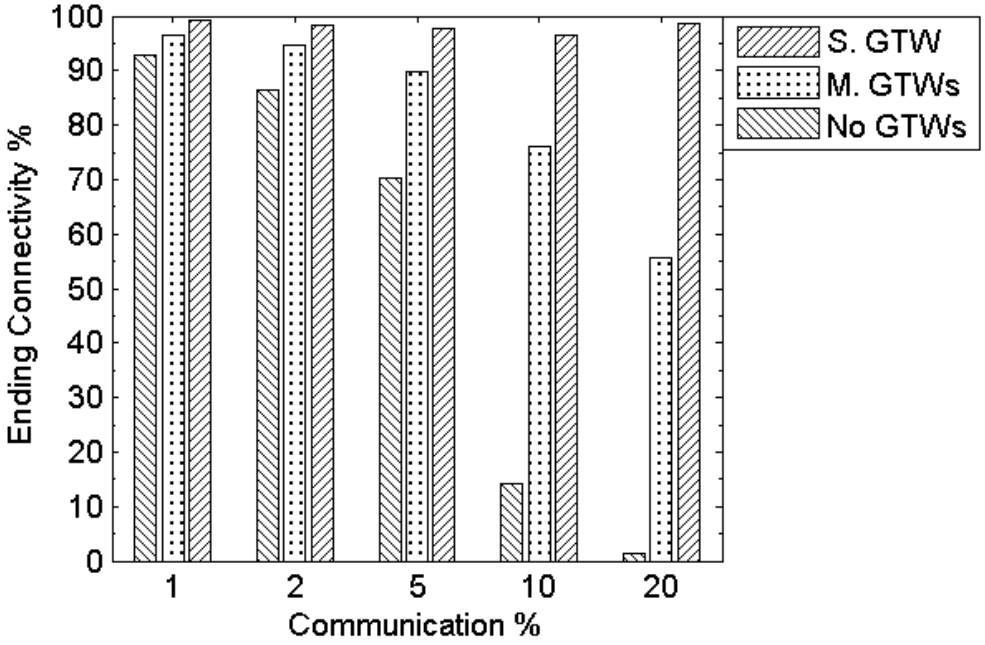


Figure 5.5: Remaining Network Connectivity for Different Communication Loads

Figures 5.4 and 5.5 present the remaining network energy and connectivity results, respectively, for the different communication loads after 5000 time periods. Single gateway (S. GTW) C-to-C packet forwarding shows to have a remarkable lower energy consumption, due to the fact that for any time period the number of sensors jumping to forward packets in the network is bound by the number of clusters without direct connectivity, thus, consuming less energy. Multiple gateways (M. GTWs) C-to-C packet forwarding consumed more energy, but proved to be more efficient than packet forwarding without gateways (No GTWs). Connectivity results show to be highly correlated with network energy for a jumping sensor topology; as more nodes deplete their energy, other dependent nodes are left without connectivity.

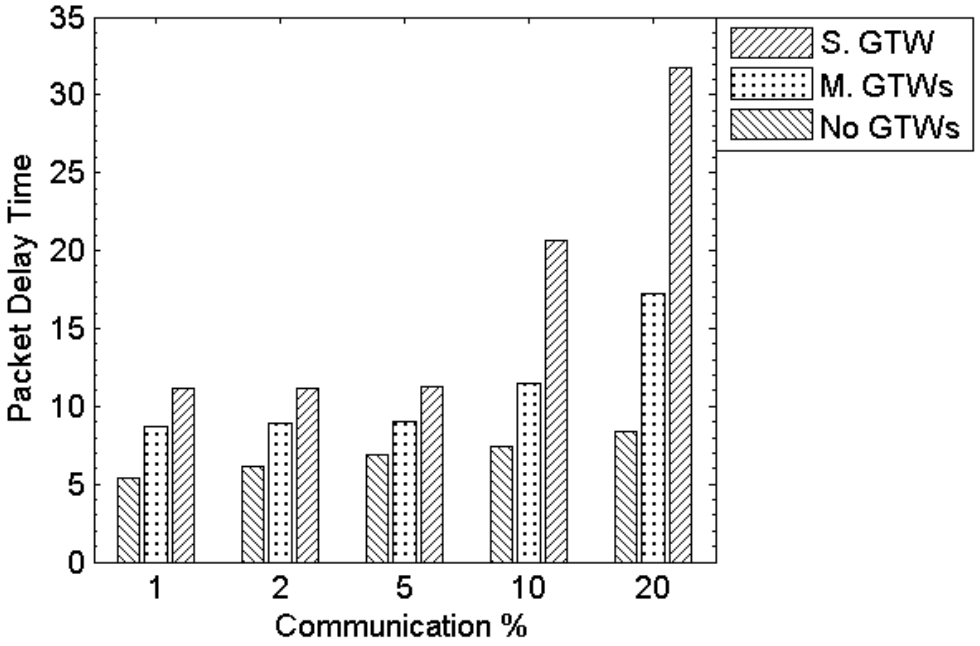


Figure 5.6: Mean Packet Delay Time for Different Communication Loads

A trade-off in energy efficiency and packet delay time is evident in Figure 5.6. Packet delay time for Single GTW showed to be constant and slightly higher than the others, for

communication loads 1%, 2%, and 5%, whereas rapidly increasing for higher communication loads, due to network saturation and bottleneck effect created from gateways. Multiple GTWs is more resilient to delay incurred with increased communication load, since it provides alternative routes to nodes within each cluster.

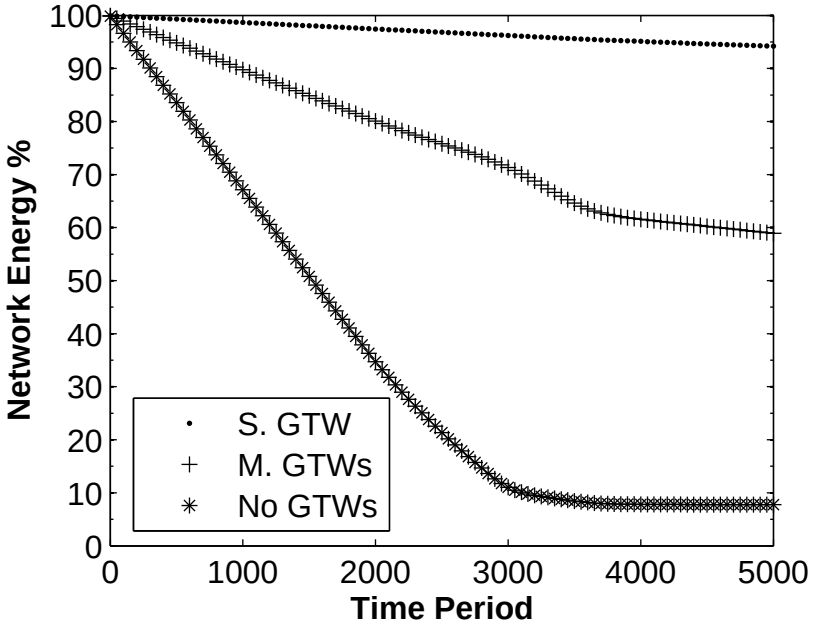


Figure 5.7: Network Energy Consumption for 20% Nodes Generating Packets

The energy consumption rates of the network is evaluated for the different communication loads. Single GTW packet forwarding shows the smallest energy consumption ratio, followed by Multiple GTWs packet forwarding. For 20% communication loads (Figure 5.7) is noted that after 3000 time periods there a is an increase in the energy consumption rate for Multiple GTWs. This change in rate is because more gateways are being used concurrently due to high packet traffic on the network. The energy consumption rate is reduced later (around 3500 time period), which is a result of the successful reconfiguration of the network to allocate new gateways. With packet forwarding without gateways (No GTWs), most of the network nodes have depleted their energy by this time. However, the network energy is not shown to reach

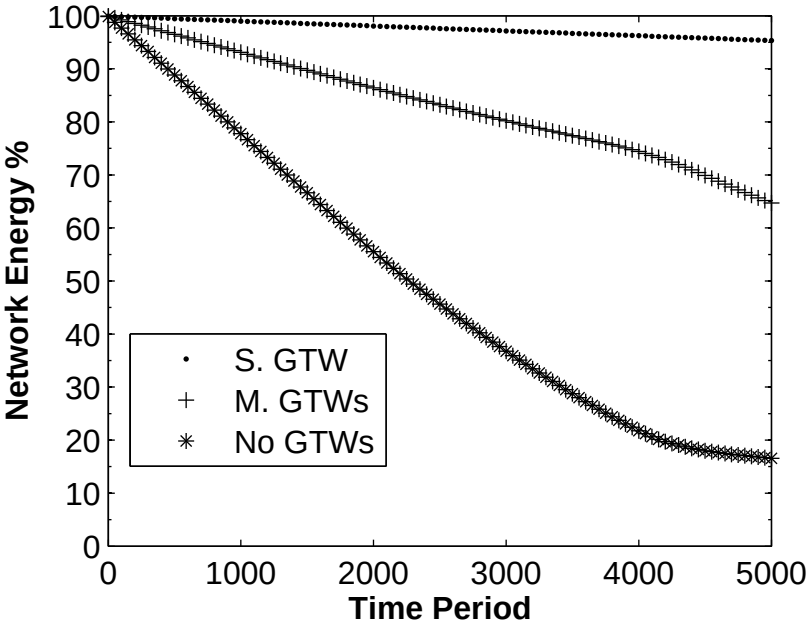


Figure 5.8: Network Energy Consumption for 10% Nodes Generating Packets

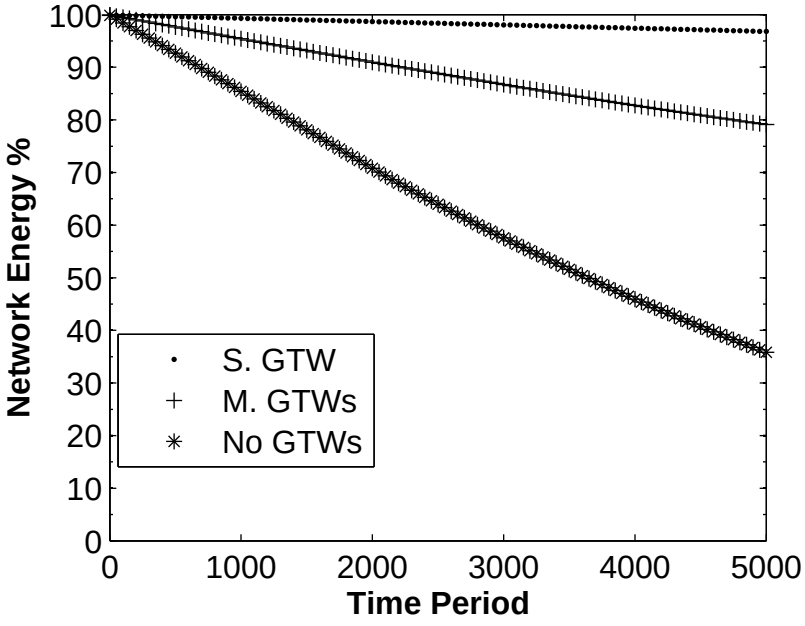


Figure 5.9: Network Energy Consumption for 5% Nodes Generating Packets

zero since for simulation purposes, a node was consider dead after reaching the threshold energy value of $5J$. A similar pattern is seen for 10% communication loads (Figure 5.8) with slightly lower energy consumption rate. The same increase in energy consumption rate is

seen again for Multiple GTWs and No GTWs, but occurring later in time (around 4250 time period) due to the less packet traffic generation. The energy consumption rates for lower communication loads (Figures 5.9, 5.10, and 5.11) were comparatively lower to the others, as expected.

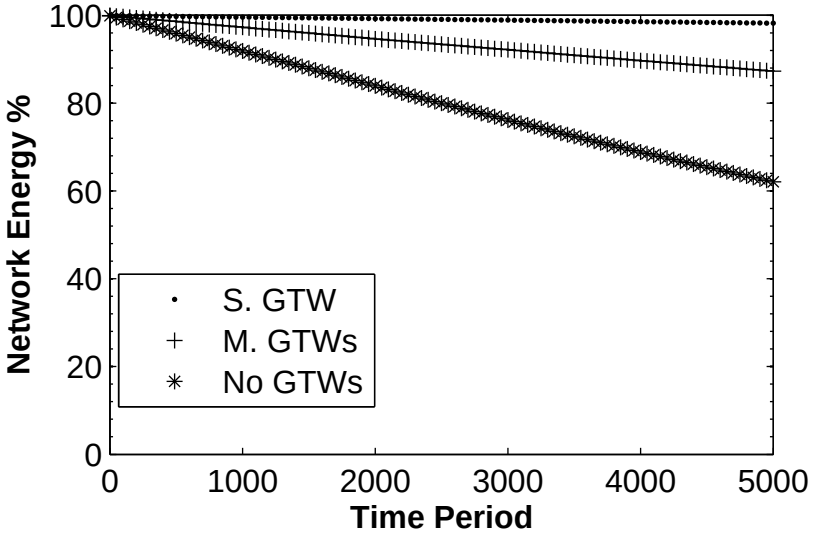


Figure 5.10: Network Energy Consumption for 2% Nodes Generating Packets]

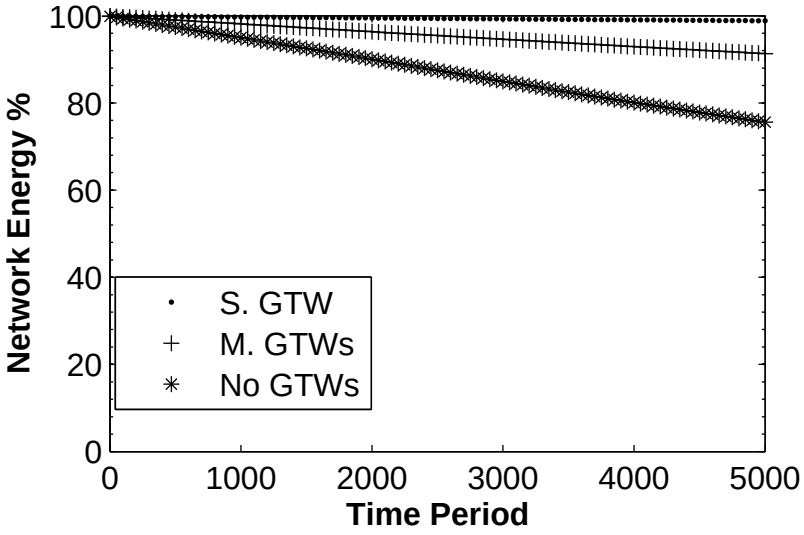


Figure 5.11: Network Energy Consumption for 1% Nodes Generating Packets

5.6 Summary

This chapter discussed the topology and operation of a jumping sensor network. It defined two decentralized algorithms for the discovery of isolated nodes, *aggressive* and *smart* discovery, to propagate base station connectivity. Both algorithms rely on the usage of jumping sensors on the boundary of a cluster, without prior topology information. The efficiency of the algorithms is studied in terms of network energy consumption and connectivity propagation. Simulation results from both algorithms showed to increase the total network connectivity to 100% for each of the evaluated topology scenarios. Also, results showed significant savings in network wide energy consumption for discovery tasks, 57% energy saving for *aggressive* discovery and 79% energy saving for *smart* discovery. Moreover, cluster to cluster (C-to-C) packet forwarding schemes relying on boundary jumping sensor gateway nodes are defined and simulated, showing to have remarkable low network energy consumption through network utility life.

Chapter 6

WIRELESS SENSOR NETWORKS AND TOPOLOGY REDEPLOYMENT

6.1 Preliminaries and Motivations

Mobile wireless sensor networks are often faced with redeployment-decision problems. Whenever a sensor node lacks persistent connectivity with neighboring nodes, a decision must be made to either relocate, increase signal strength, or redeploy sensor nodes as an attempt to increase network connectivity. Wireless sensor topologies usually start using node's maximum transmission power during network deployment and setup, Ramanathan and Rosales-Hain (2000). However, network connectivity issues persist mostly because of power-limited communication radios in the sensor nodes and the presence of line-of-sight obstacles reducing effective communication ranges.

Although specialized types of sensor motes, i.e., jumping sensors, can enhance network connectivity, in order to have a functional network it is important to address connectivity issues with respect to the application to be served. For instance, network intensive applications, such as mission-critical security surveillance, might require persistent connectivity

among sensor nodes. In such applications, redeployment of sensors is an alternative to sustain the networking demands. On the other hand, some applications can withstand intermittent connectivity (delay tolerant), meaning that communications between node pairs do not require long lasting wireless communication links. Instead, the communication occurs for short periods of time during the network lifespan. For delay tolerant applications, the network cost for redeployment could exceed the communication gains provided by jumping sensors.

The network connectivity metric is not enough to represent the network performance potential, i.e., a network evaluation indicative of network suitability for application's networking requirements. For example, a network performance metric score should reflect the existence of intermittent (non-persistent) vs. persistent communication links between nodes and base station, e.g., the more nodes connected to the base station via intermittent links, the lower the metric score and the less suitable for network intensive applications. Therefore, an evaluation metric representing the fitness of a WSN topology in terms of the quality of its connectivity is required.

This research is focused on analyzing network topologies established with the use of jumping robots sensors, and to defining a Quality of Connectivity (QoC) network performance metric. Moreover, since efficient node redeployment is a key aspect for enduring sensor networks, the QoC is adapted as an optimization metric for node redeployment. Redeployment-decision algorithms with different approaches to enhance network performance are presented. A discussion of the obtained results from evaluations and performance comparisons is provided.

6.1.1 Chapter Organization

First, Section 6.2 provides relevant definitions and outlines network connectivity derived problems. Second, Section 6.3 is dedicated to the discussion of jumping sensor airborne communication and network performance aspects from the application perspective. Third, in Section 6.4 we describe the network topology as a network fitness formula considering nodes' quality of connectivity. Fourth, Section 6.5 discusses cluster merging algorithms aimed to maximize network performance. Fifth, Section 6.6 presents results from evaluations and performance comparisons for the proposed algorithms. Finally, Section 6.7 summarizes and concludes this chapter.

6.2 System Model

In this chapter we consider single sink node topologies where sensor nodes collect field readings and forward them in communication packets to a sink node. We define a node's connectivity as the ability to communicate with a sink node, directly or through multi-paths formed by neighbor nodes. For the remainder of this section we refer to the sink node as the base station node.

Node's connectivity is one of the factors having the biggest impact on the network performance. If a node cannot forward the field readings to the base station, then it has to wait until a communication path becomes available. Newer communication paths become available as sensor nodes get discovered by others. Depending on the application requirements, field readings might become outdated by the time a path to the base station is discovered. Hence, it is imperative to provide connectivity to as many sensor nodes as possible.

Even though connectivity is crucial for network performance, other factors have to be

taken into account when evaluating a network topology. For instance, a fragmented topology will impact network responsiveness and the capacity to meet application's flow demands. On the other hand, bad network configuration and bad reallocation decisions are likely to deteriorate network utility life.

6.2.1 Connectivity and Sensor Clusters

Network connectivity is evaluated as the ratio of nodes having connectivity with the sink node out of the total number of deployed nodes. Consider a network represented as a simple graph N with $|N|$ nodes. We define the nodes with network connectivity as those that belong to the connected component containing the base station. Call this connected component C_B, and $|C_B|$ the number of nodes having connectivity. Hence, network connectivity can be defined as Eqn. (6.1).

$$Network\ Connectivity = \frac{|C_B|}{|N|} \tag{6.1}$$

The connectivity of a node depends on the existence of communication paths to the base station formed by sensor nodes. Sensor nodes that are within their wireless communication radius establish link pairs to form communication paths. However, these link pairs are not always persistence, mostly due to their wireless nature, which creates an impact in node's connectivity.

There are a number of scenarios that can disrupt wireless communication links, two of them are: long distances between node pairs, and obstacles in the network topology. It is known that wireless signal strength fades with the increase of the propagation distance. When nodes are randomly deployed in the field, e.g., air-dropped Song *et al.* (2009), there is hardly any control on their initial distribution. The problem with this is that some nodes

end up too far apart from each other, thus "excommunicating" them from the network. On the other hand, the presence of obstacles can cause disruption in the signal propagation, and in some scenarios prevent mobile nodes from getting closer, e.g., river streams, lakes, boulders, etc.

It is common for a network topology to start divided in clusters of nodes. We define a **cluster** as a subset of one or more nodes from the network having stable (long lasting) communication links. In other words, by removing intermittent links from the network, the resulting graph's connected components are exactly the clusters. Many clusters are composed of sensor nodes distant from the base station. Naturally, in a single base station topology any cluster, other than the one where the base station is, will have intermittent connectivity or no connectivity all. Therefore, the number of clusters in the network can have an impact in the overall connectivity, i.e., persistent (desired in rapid response applications) vs. intermittent (good for delay tolerant applications).

6.3 Network Performance

We wish to obtain the highest performance possible from a sensor network topology. This section is dedicated to the discussion of jumping airborne communication and network performance aspects from the application perspective.

6.3.1 Jumping Airborne Communication

Jumping sensor robots allow relocation in rugged terrains where conventional wheeled sensor robots cannot perform. In addition, jumping sensors while airborne can establish communication links between distant sensor nodes. This is possible because the change in altitude

clears the communication line-of-sight allowing the transmitted signal to propagate farther, thus, enabling communication with distant receivers Cintrón *et al.* (2009, 2012).

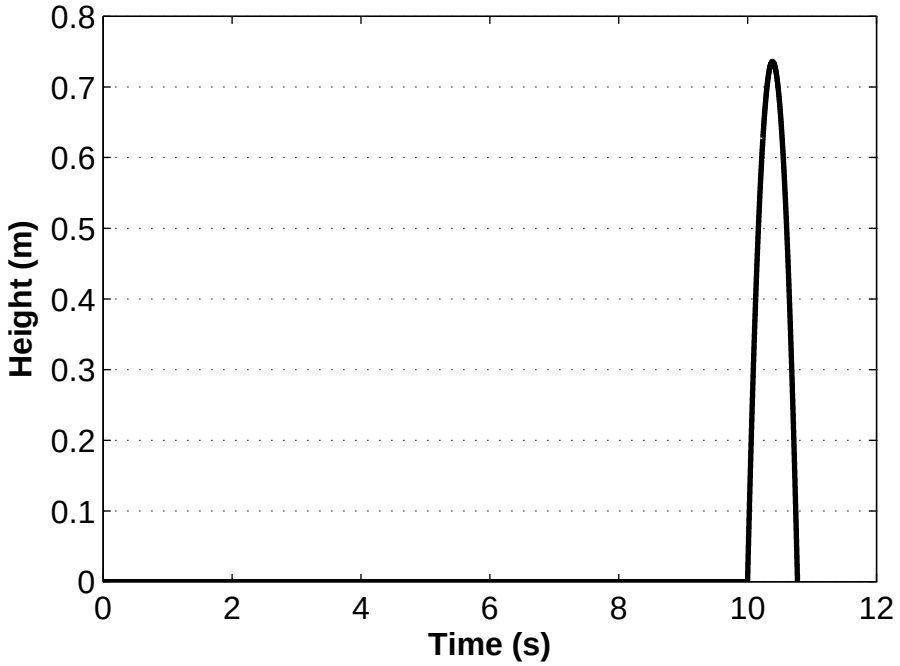
The availability of airborne communication links is dependent on the time window (T_c) available to communicate during the total time of flight, which is dependent on the threshold height (H_{Th}) the sensor must reach for communication and the maximum jump height (H_{max}) that it can reach. This relation is expressed in Eqn. (6.2) where g is the gravitational acceleration.

$$T_c = 2\sqrt{\frac{2(H_{max} - H_{Th})}{g}} \tag{6.2}$$

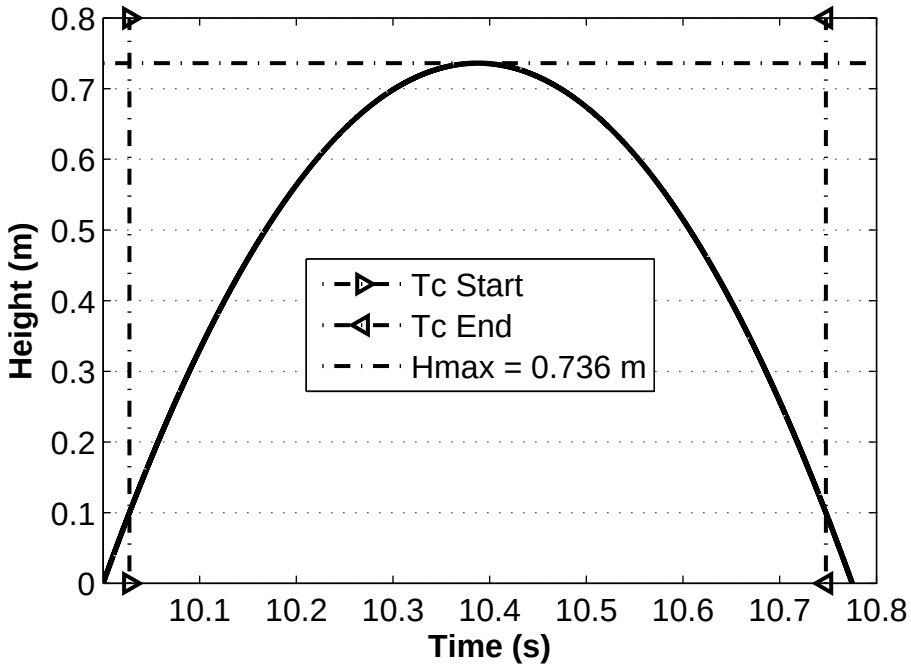
Additionally, we have to consider how long it takes for a jumping robot to perform a complete jump cycle, Eqn. (6.3). This includes any time required for the jumping robot to prepare before a jump (e.g., loading of jumping mechanism, self righting and positioning), and the time of flight obtained from the jump.

$$T_{cycle} = T_{pre-jump} + T_{ToF} \tag{6.3}$$

For purpose of analysis we consider the MSU Jumper, a miniature steerable jumping robot prototype presented by Zhao et al. Zhao *et al.* (2013). The prototype has a jumping loading time of 10 seconds, with an estimated normalized jump height (height reached from vertical jumping) of 87 cm and 73.6 cm when carrying 4 grams and 8 grams of payload weight, respectively. Figure 6.1 plots both, the jump cycle as a function of time in Fig. 6.1a, and a zoom-in of the time of flight ($T_{ToF} = 775$ ms) when jumping to a height of 73.6 cm in Fig. 6.1b. The communication time window ($T_c = 720$ ms) is displayed for a threshold height of 10 cm.



(a) Jump Cycle



(b) Communication Time (T_c) Above 10 cm Threshold Height

Figure 6.1: Jumping Robot Jump Cycle and Time of Flight for 73.6 cm Jump Height

From Fig. 6.1a, it can be observed that the communication time (T_c) is small relative to the entire jump cycle time (T_{cycle}) for this particular jumping robot prototype when carrying a load weight of 8 grams. The jump cycle time can be a determining factor for the application's suitability for airborne sensor communication with jumping sensors.

Let us consider a single jump for communication, then the maximum transmission rate during a jump will be equal to the link speed R_{Tx} obtained from the employed wireless protocol. However, this rate is conditioned to the communication time window during the jump. Therefore, it is practical to compute the number of communication frames that can be transmitted during a jump. This can be obtained as follow:

$$t_{frame} = \frac{F_{bits}}{R_{Tx}} \tag{6.4}$$

$$F_{jump} = \frac{T_c}{t_{frame} + t_{delay}} \tag{6.5}$$

where,

t_{frame} time it takes to transmit a frame.

t_{delay} inter frame delay time for synchronization between transmitter and receiver.

F_{bits} number of bits in a communication frame.

R_{Tx} transmission rate (link speed) in $bits/sec$.

F_{jump} maximum number of frames that can be transmitted per jump.

For instance, using the following networking parameters: frame size of 25 Bytes (10 Bytes Payload, 6 Bytes physical layer and 9 Bytes MAC sublayer 802.15.4-IEEE-Standard (2006)),

R_{Tx} = 250 Kbps, t_{delay} = 7.5 ms, and assuming airborne communication can occur with same jump height values as presented in Fig. 6.1b for a T_c = 720 ms. Replacing these values in Eqn. (6.4) and (6.5), the maximum number of frames that can be transmitted per jump is 86.77 $[frames/jump]$.

Although the communication time is small when compared to the jump cycle time, it is enough for short communications. On the other hand, communication that occur with high frequency or exceeding the available communication time per jump can suffer from delay.

The average frame rates for ground surface (R_{Gx}) and airborne communication (R_{Jx}) with jumping sensors can be obtained with Eqn. (6.6) and (6.7), respectively.

$$R_{Gx} = \frac{1}{t_{frame} + t_{delay}} \tag{6.6}$$

$$R_{Jx} = \frac{F_{jump}}{T_{cycle}} \tag{6.7}$$

Considering the previous networking parameters, an average frame rate for sensor within communication range at ground surface level is found to be R_{Gx} = 120.48 $[frames/sec]$. However, the average frame rate for airborne communication is found to be R_{Jx} = 8.05 $[frames/sec]$. This lower frame rate is caused by the time it takes for a jumping robot to complete a jump cycle. The jumping cycle causes packet transmissions to occur in bursts, resulting in increased network communication latency. It should be noted that these rates are for the specified jumping characteristics of the MSU Jumper, which is designed to maximize both jumping height and travel distance.

6.3.2 Gateway Airborne Communication

Jumping sensors topologies can employ jumping capable sensors as gateway nodes to disconnected clusters, Cintrón and Mutka (2010). The jumping frequency of a gateway jumping sensor is determined by the packet forwarding rate the node must achieve. This is related to the packet arrival rate, which is application dependent. To determine the viability of gateway jumping sensors for a particular application, it is necessary to predict the total packet rate R_{Ci} for clusters.

$$R_{Ci} = r_i * n_i \tag{6.8}$$

where,

R_{Ci} the total packet rate for cluster i.

r_i the average packet output rate for sensing nodes in cluster i.

n_i the number of active nodes in cluster i.

If the jumping node can buffer and transmit P_{jump} packet in a jump, then it can wait a maximum of t_w before having to forward the corresponding data packets. This relation is expressed below.

$$t_w = \frac{P_{jump}}{R_{Ci}} \text{ seconds} \tag{6.9}$$

Moreover, given an application messaging frequency, the number of sensor nodes a jumping sensor gateway can support can be determined with Eqn. (6.8) and (6.9).

Case Study I: Consider an environmental sensing application performing soil moisture monitoring every 10 minutes ($r_i = 0.0017$ $[pck/s]$) as presented by Wu and Liu Wu and Liu (2012). Each jumping sensor having to wait 10 seconds to perform a jump, and a $P_{jump} = 86$ $[pck/jump]$. Solving Eqn. (6.9) for the number of nodes, we obtain:

$$n_i = \frac{P_{jump}}{r_i * t_w} \tag{6.10}$$

$$n_i = \frac{86}{0.0017 * 10} \approx 5058 \text{ nodes}$$

In other words, to prevent buffer overflow, under the above mentioned networking characteristics, a single jumping sensor gateway could serve up to 5058 sensing nodes.

Case Study II: In a Moving Object Tracking application, the monitored area is typically divided into sensing regions Lin *et al.* (2006). Event messages are sent when an object exits or enters a sensing region. The event message is a short (about 10 Bytes) message containing a time stamp, object ID (identifier), exiting region ID, and entering region ID. For continuous monitoring all event messages are forwarded to the base station. We now consider an example where a sensor network tracks people running in an area using sensors with a sensing radius of 5 $[m]$. A runner with a speed of 20 $[Km/hr]$ will have an area transition rate of 0.56 $[Transitions/sec]$. This results in an event message rate of 1.12 $[msg/sec]$. While the average ASC frame rate $R_{Jx} = 8.05$ $[frames/sec]$ would support the ability to track 7 objects simultaneously, the latency introduced by the jump cycle results in outdated position information. However, this is not a problem for applications studying animal behavioral patterns, not requiring position information in real-time, since the data is aggregated for later evaluation Dyo *et al.* (2010).

It has been shown that the number of sensor nodes, the packet rate and delay tolerance from the application are key factors to determine whether ASC is going to be able to sustain the communication load of a cluster. Moreover, jumping sensors have limited power, hence ASC should be used to provide temporary communication while topology redeployment takes place to establish long lasting communication links. The network redeployment process is discussed in the next sections.

6.4 Quality of Connectivity

Network connectivity measures the ratio of nodes able to connect to the base station. However, with this piece of information alone we cannot say much about how the network will perform under different scenarios, or if it is adequate for a particular application. Hence, we need a measurement to describe the type of connectivity the network can provide in its current topology state.

The Quality of Connectivity (QoC) evaluates the network connectivity ratio in terms of how segregated and distributed are the sensor nodes with connectivity in the topology. In principle, cluster to cluster communication suffers from intermittent links. The higher the number of these intermittent communication links on a path to base station, the higher is the expected delay for a message to arrive at the base station. Therefore, QoC looks at the entire network topology as levels of connectivity.

The connectivity level of a cluster can be determined with the number of intermittent links a message must traverse to arrive at the base station. The cluster containing the base station is considered the first level. One level up are those clusters having at least one intermittent communication link to the first level. Levels above depend on their immediate

lowest level to acquire connectivity, as shown in Fig. 6.2.

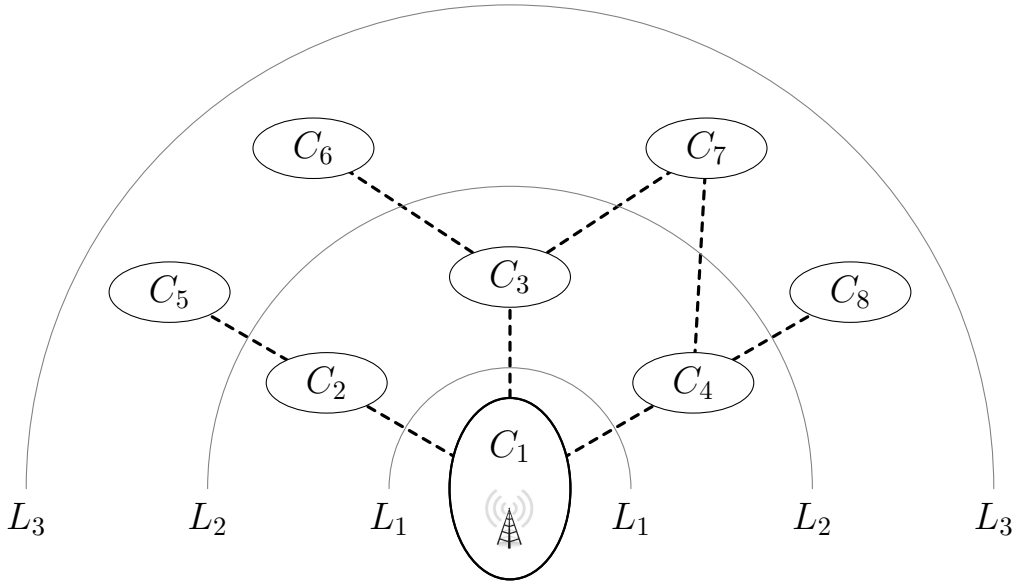


Figure 6.2: Single base station clustered topology depicting connectivity levels

To be able to obtain a score for a network topology we have to assign a scoring contribution weight value to each level on the topology. Lower levels are desired, since they are expected to have lower delay than higher up levels. Therefore, they should contribute more to the overall network score. This is achieved by assigning a weight that decreases as the level increases. Also, the level contribution to the QoC has to be proportional to the number of nodes on it.

We use a geometric distribution to assign diminishing weights to each connectivity level away from the base station, Eqn. (6.11).

$$\lambda_i = r^i \tag{6.11}$$

In order to form a probability distribution we chose a common ratio, $r = \frac{1}{2}$. This satisfies:

$$\lim_{k\to\infty} \sum_{i=1}^{k} r^i = \frac{r}{1-r} = 1 \tag{6.12}$$

This probability distribution will be applied to the connectivity levels. We define the connectivity levels to be subsets of C_B such that:

$$C_B = \bigcup_{i}^{\ell} L_i \tag{6.13}$$

where,

$$|C_B| = \sum_{i=1}^{\ell} |L_i| \tag{6.14}$$

There are ℓ connectivity levels indexed as $L_1, \ldots, L_\ell$. Each L_i contains $|L_i|$ nodes. The geometrically weighted contribution of each level is represented as:

$$\sum_{i=1}^{\ell} \lambda_i |L_i| = \lambda_1 |L_1| + \lambda_2 |L_2| + \cdots + \lambda_\ell |L_\ell| \tag{6.15}$$

In order to provide a representative score of how fragmented the network topology is, an average over the number of connected nodes is computed. However, because the number of connectivity levels is finite, the sum of the weights is less than one. For this reason, we normalize each weight and take the average using:

$$|C_B| \sum_{i=1}^{\ell} \lambda_i \tag{6.16}$$

This introduces the network QoC score, Eqn. (6.17). The QoC score is near one (zero) if and only if the nodes having connectivity are distributed in few (many) clusters. A score value

of one is equivalent to having all the nodes with connectivity in a single cluster network.

$$QoC = \frac{\sum_{i=1}^{\ell} \lambda_i |L_i|}{|C_B| \sum_{i=1}^{\ell} \lambda_i} \tag{6.17}$$

6.4.1 QoC Enhancement

One of the network managing tasks is to identify node's actions that can increase or reduce the overall performance. For example, the decision of relocating a set of nodes to merge two clusters. Such operation creates an impact in network expected life as the sensor nodes consume more energy while relocating, but can potentially increase network performance. Therefore, we want to evaluate the potential performance gain before any cluster merging operation.

Network QoC allows to evaluates the network topology on its current state. The contribution to the QoC from each level depends on the number of nodes on the level, and the contributions from levels above them. Hence, a merging operation between two clusters in different connectivity levels represents a QoC increase. This is true because the merging operation increases the lower level size, and brings dependant clusters from above levels one level closer to the base station.

A cluster contribution to the QoC can be represented as a recursive function QoC_i^c, presented in Eqn. (6.18),

$$QoC_i^c = \lambda_i(|L_i^c|) + QoC_{i+1}^c \tag{6.18}$$

where i indicates the current connectivity level the cluster c is at, $|L_i^c|$ the size of the level considering only those dependent clusters of c. The recursion will stop at the last dependant

connectivity level of cluster c, where $QoC_{i+1}^{c} = 0$. The QoC_{i}^{c} function results in a rational number that has to be normalized before it can be used to evaluate which cluster merge operation has the greatest impact on the QoC.

Since we are using a geometric distribution (Eqn. (6.11)) to assign the connectivity levels weights λ_i, we can simplify Eqn. (6.18) to represent all level contribution as:

$$Q_{i}^{c} = \lambda_1(|L_{i}^{c}| + Q_{i+1}^{c}) \tag{6.19}$$

Then, the QoC_{i}^{c} contribution for $i \geq 2$ is obtained by:

$$QoC_{i}^{c} = \lambda_{i-1}(Q_{i}^{c}) \tag{6.20}$$

Equations (6.19) and (6.20) allow for a distributed computation of each cluster contribution, thereby reducing the computational load on a single cluster.

To better understand the process of using QoC as a decision factor for cluster merging, let us consider the example topology presented in Fig. 6.3. Clusters' sizes for the example topology are indicated in Table 6.1.

Table 6.1: Cluster's Sizes and Distribution by Connectivity Level of Network Topology in Fig. 6.3

L_i	L_1	L_2		L_3			L_4
C_j	C_1	C_2	C_3	C_4	C_5	C_6	C_7
$\|C_j\|$	2	15	21	30	8	4	20
$\|L_i\|$	2	36		42			20

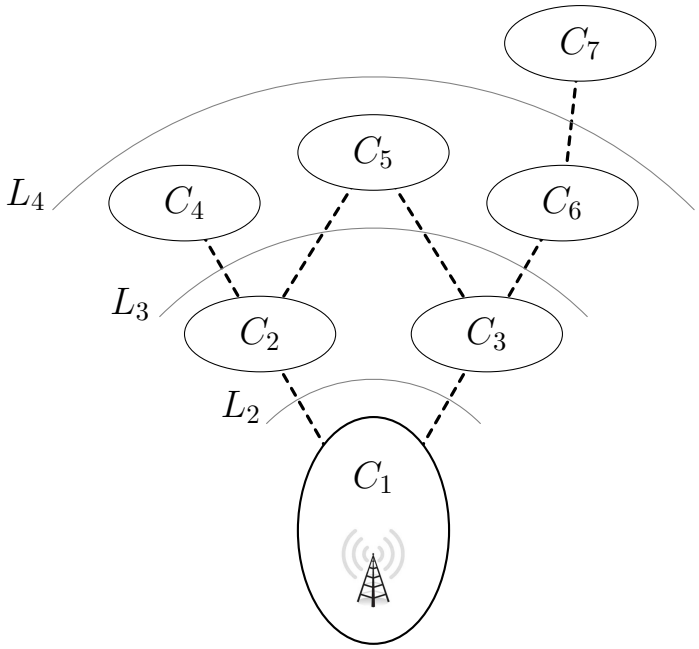


Figure 6.3: Network Topology Example for QoC_2^* Contribution Evaluation

Using the values from Table 6.1 in Eqn. (6.17) we can obtain the QoC of the topology on its current state.

$$\begin{aligned} QoC &= \frac{\lambda_1(2) + \lambda_2(36) + \lambda_3(42) + \lambda_4(20)}{100 \sum_{i=1}^{4} \lambda_i} \\ &= 0.176 \end{aligned}$$

We want to evaluate which cluster, C_2 or C_3, merging with C_1 will result in the highest QoC increase. For this particular scenario, the number of nodes that will benefit from either merge operation is equal, which is:

$$|C_2| + |C_4| + |C_5| = |C_3| + |C_5| + |C_6| + |C_7| = 53 \tag{6.21}$$

This was selected to demonstrate that the QoC is not only impacted by the number of nodes,

but also by how they are distributed in the topology.

Evaluating both cluster's QoC_i^c contributions, QoC_2^2 and QoC_2^3, we obtain:.

$$\begin{aligned} QoC_2^2 &= \lambda_2(15) + \lambda_3(38)) \\ &= 8.5 \end{aligned} \tag{6.22}$$

$$\begin{aligned} QoC_2^3 &= \lambda_2(21) + \lambda_3(12) + \lambda_4(20) \\ &= 8.0 \end{aligned} \tag{6.23}$$

The contributions have to be normalized in order for cluster C_1 to compare them. We normalize each QoC_i^c contributions by the sum of connectivity weights from the resulting graph depth, should a merge with the cluster takes place. Let us call this the gain $G_i^{a,k}$ of dependent cluster C_k merge with anchor cluster C_a, and ℓ_k the depth of the resulting graph.

$$G_i^{a,k} = \frac{QoC_{i+1}^k}{\sum_{i=1}^{\ell_k} \lambda_i} \tag{6.24}$$

Evaluating the gain for each cluster merge, we obtain:

$$G_1^{1,2} = \frac{8.5}{\sum_{i=1}^{4} \lambda_i} = 9.07 \tag{6.25}$$

$$G_i^{1,3} = \frac{8.0}{\sum_{i=1}^{3} \lambda_i} = 9.14 \tag{6.26}$$

Hence, the merge between cluster C_1 and C_3 will provide the higher increase to the QoC in the network. This can be verified by computing the QoC after the merge. By merging clusters C_1 and C_3, the QoC of the network topology is increased to:

$$\begin{aligned} QoC &= \frac{\lambda_1(2+21) + \lambda_2(15+8+4) + \lambda_3(30+20)}{100 \sum_{i=1}^{3} \lambda_i} \\ &= 0.28 \end{aligned}$$

Instead if we chose to merge clusters C_1 and C_2, the increase in the QoC of the network topology was going to be less:

$$\begin{aligned} QoC &= \frac{\lambda_1(2+15) + \lambda_2(21+30+8) + \lambda_3(4) + \lambda_4(20)}{100 \sum_{i=1}^{4} \lambda_i} \\ &= 0.2666 \end{aligned}$$

It is worth noting that even with the individual QoC_i^c contributions favoring the merge of clusters C_1 and C_2, the impact from reducing a whole connectivity level by merging clusters C_1 and C_3 is taken into account when QoC_i^c values are normalized in Eqn. (6.24).

6.5 Cluster Merging Decision Algorithms

Cluster merge operations imply relocation, or even additional deployment, of sensing nodes to establish persistent communication links between clusters. Once the persistent commu-

nication links are established, the involved clusters become unified into a single cluster. Persistent communication links are desired to enhance network connectivity. The network QoC performance metric takes into account the existence of intermittent links, since they tend to cause delay in network communications.

This section presents different algorithms for merging clusters in mobile sensor network topologies. The merging of clusters in the network topology requires either, node relocation or redeployment. The scope of this section is to establish the order of merging operations to occur in the network topology. The QoC is suited for this purpose. The algorithms goal is to maximize the QoC of the network by smartly choosing clusters to perform merging operations. The cluster merging decision algorithms are divided into centralized and distributed categories, Section 6.5.1 and Section 6.5.2, respectively. Furthermore, the algorithms differ mainly in the criteria used to select a merging candidate, and the cluster's subset used for candidate selection (the merging candidates' scope). All the algorithms terminate when there are no more cluster merge options.

There are three merging selection criterion, namely:

- **Maximum QoC yield (MQY):** Cluster merging action that yields maximum QoC in the network topology.
- **Largest cluster size increase (LCSI):** Cluster merging action that results in the largest cluster size increase.
- **Random:** Random selection of cluster merging action. Used as a base performance comparison.

Each algorithm is characterized by a unique combination of the merging criteria, scope, and computing approach (centralized or distributed). An overview of the algorithms' charac-

Table 6.2: Algorithms' Characteristic Overview

	Algorithm	MQY	LCSI	Random	Scope
Centralized	BMN	✓			All
	BMN-BS	✓			L_1
	LIMN		✓		All
	LIMN-BS		✓		L_1
	RMN			✓	All
	RMN-BS			✓	L_1
Distributed	DBMN	✓			All
	DBMN-BS	✓			L_1
	S-DBMN	✓			L_1, L_2
	DLIMN		✓		All
	DRMN			✓	All

teristics is provided in Table 6.2. The scope of the algorithms is expressed as the connectivity level where cluster's merging evaluations are performed. For example, an L1 scope indicates merging operations between level 1 and 2 are allowed, whereas a scope of both, L1 and L2, indicates merging between level 1, 2, and 3 are allowed. Detailed descriptions for the algorithm are provided in Section 6.5.1 and 6.5.2.

6.5.1 Centralized Approaches

The centralized approaches rely on a common reachable entity, such as the base station, to collect data from sensor nodes, perform analysis, and make decisions based on an overall view of the network topology status. Decisions are taken with the objective to create a global improvement in the network QoC.

6.5.1.1 BMN and BMN-BS

Best Merge Next (BMN) selects the cluster pair to perform the merging operation, resulting in the maximum QoC yield. BMN-BS in contrast, only evaluates candidate clusters that

are neighbors to the base station cluster, and selects the one with maximum QoC yield to perform the merging operation. BMN can be considered as a brute-force approach to find the sequence of cluster's merging operations yielding the highest improve in QoC.

6.5.1.2 LIMN and LIMN-BS

Large Increase Merge Next (LIMN) selects the cluster merging operation that results in the largest cluster's size increase. LIMN-BS in contrast, only evaluates candidate clusters that are neighbors to the base station cluster, and selects the one having the largest cluster size to perform the merging operation.

6.5.1.3 RMN and RMN-BS

Random Merge Next (RMN) and Random Merge Next from base station (RMN-BS) are the study base-cases for performance comparison. RMN, randomly selects cluster pairs in the network topology to perform merge operations. RMN-BS, randomly selects cluster neighbors to the base station cluster to merge.

6.5.2 Distributed Approaches

In contrast to the centralized approaches, the distributed approaches work by applying the merging selection criterion locally to each cluster. Any clusters having dependent[1] clusters communication links will perform the merging selection process, by applying the selection criteria of the algorithm employed. Due to the distribute nature of the approaches, and as a measure to limit any possible control communication overhead, there is no particular

[1]Dependent clusters are those clusters at farther connectivity levels from base station that depend on other clusters routing capabilities to be connected to the network, see Section 6.4.

order of precedence for cluster merges to occur. Hence, cluster merge operations can occur simultaneously by default.

6.5.2.1 DBMN, DBMN-BS, and S-DBMN

Distributed Best Merge Next (DBMN) and the other variants are similar to BMN in the sense that they seek to make a direct impact on network QoC by selecting the maximum QoC yield merging operations on each cluster having dependents. However, since these approaches are designed to be carried in a distributed manner, and the QoC is an overall (global) network performance measurement not available to all the clusters, each cluster computes the individual Gain as presented in Section 6.4.1.

The difference between distributed variants is in the cluster's subset used for candidate selection. In DBMN all the cluster having dependent clusters execute merging operations. In DBMN-BS, only merge operations between the base station cluster and its neighbors are allowed. Special DBMN (S-DBMN) allows base station clusters neighbors to execute merging operations with their dependents as well. In other words, merge operations in DBMN-BS can occur only between connectivity level[2] 1 and level 2, while in S-DBMN merge operations are allowed between connectivity levels 1, 2 and 3.

6.5.2.2 DLIMN

Similar to LIMN, Distributed Large Increase Merge Next (DLMN) uses largest cluster's size increase to select among the candidate clusters. The main different is that all clusters having dependent clusters will employ the selection procedure locally. In other words, clusters with dependents select the local LIMN, and proceed with the merging procedure.

[2]Connectivity level 1 has only one cluster with the network base station.

6.5.2.3 DRMN

Similar to RMN, in Distributed Random Merge Next (DRMN) each cluster with dependents executes a random selection to perform merge operations.

6.6 Evaluations

Simulations were performed to evaluate the performance of the cluster merging decision algorithms. First, a detail description of the generated simulation environments is provided. Second, obtained results are presented and evaluated.

6.6.1 Simulation Environment

Network topologies were generated and evaluated using a custom simulator in Matlab. Variable inputs were established for the generation of network topologies: number of connectivity levels ℓ, number of clusters, and the total number of sensors nodes. The number of clusters was selected to be five (5) times the number of connectivity levels, and the total number of sensor nodes was selected to be twenty (20) times the number of clusters. All the algorithms were evaluated under different number of connectivity levels. Topologies with three (3), nine (9), and fifteen (15) connectivity levels were found to be representative for all the other evaluated topology configurations.

The following topology setting requirements were established:

- A connectivity level by definition must have at least one cluster.
- A cluster must be composed of at least one sensor node.
- Every cluster must have at least one communication link with a cluster at a lower

connectivity level (with the exception of the Base Station cluster, since by definition, it is the first and lowest existing connectivity level).

Sensor nodes were randomly distributed into clusters; the clusters were randomly assigned to different connectivity levels. Communication links between clusters were classified as *inter-level* (connections between clusters at the same connectivity level), and *intra-level* (connections between clusters at different connectivity levels). Aside from the required communication links for each cluster, additional links were randomly established with biased probability. Intra-level links had 40% chance of occurrence, while inter-level links had 20% chance.

Following the aforementioned settings, ten (10) topologies were generated for each fixed number of connectivity levels, totaling thirty (30) network topologies.

6.6.2 Evaluation and Results Analysis

The performance of the cluster merging algorithm was analysed with two metrics:

- Impact to network QoC for each cluster merge operation.
- Number of candidate evaluations associated with each cluster merger operation.

Mean values for QoC and the number of evaluations per cluster merge operation were computed for centralized and distributed approaches. All distributed algorithms, and the algorithms employing random selection functions, such as, RMN, RMN-BS, DRMN, were run ten (10) independent times for each topology to obtain their performance's mean value.

For legibility purpose, plots of the results are presented in Section 6.6.3. The results are grouped by the topology's initial number of connectivity levels (3, 9, 15). Each connectivity level has result's plots for QoC improvement versus the number of cluster merge operations,

and the number of evaluations per cluster merger operations. Results are further organized accordingly to the decisions algorithm approach, i.e., centralized and distributed.

6.6.2.1 BMN, BMN-BS, and DBMN-BS

BMN, BMN-BS, and DBMN-BS produced the maximum QoC yield per merge operation. Their difference is on the number of evaluations required before each merge operation decision. BMN analyzes each possible merge operation in the network topology, while BMN-BS and DBMN-BS only consider merge operations directly with the base station cluster. Even though the number of comparison evaluations is the same for BMN-BS and DBMN-BS (see plots in Section 6.6.3), they differ in the selection process and candidate evaluation function.

BMN and BMN-BS evaluate each candidate merge operation individually by peeking into the future and recording the future QoC yield (should that candidate be selected), a feasible procedure in less resource constrained nodes. On the other hand, in DBMN-BS each merge candidate cluster computes its expected QoC contribution and provides it to the base station cluster, who computes and evaluates the candidates' QoC gains. DBMN-BS is less computing intensive, and proves to have equal performance as its centralized counterpart.

6.6.2.2 LIMN, LIMN-BS, and DLIMN

The QoC yield per merge operation from LIMN and LIMN-BS was the closest to that obtained by MQY algorithms. Their QoC maximization strategy relies in choosing the largest cluster size increase (LCSI). The disadvantage with these approaches is that they focus only in cluster's size, neglecting the current level of connectivity of the clusters. The level of connectivity is heavily weighted by the QoC function. This is because the connectivity level of a cluster is highly correlated with the expected delay for any communication to reach

the base station.

Nonetheless, LIMN is shown to produce a higher QoC yield to that of LIMN-BS due to the bigger scope of candidate clusters, Fig. 6.4a, 6.6a, and 6.8a. The higher QoC yield from LIMN comes with a price, a greater number of comparison evaluations, making it more computing intensive at early stages of redeployment, Fig. 6.5a, 6.7a, and 6.9a. LIMN-BS, having less number of comparison evaluations, obtained a QoC yield slightly below of that from LIMN for topologies with few connectivity levels. However the difference is more noticeable in topologies having greater number of connectivity levels, Fig. 6.8a.

DLIMN, the distributed counterpart of LIMN, produced a slower increase in QoC, Fig. 6.4b, 6.6b, 6.8b. Its QoC yield is just under that obtained with DBMN, with a slight lower number of evaluations per merge operation. This is more noticeable in topologies having greater number of connectivity levels, Fig. 6.9b.

6.6.2.3 S-DBMN and DBMN

S-DBMN came in second place for QoC yield, with a low number of evaluation per merge operation, when compared to other distributed algorithms, Fig. 6.4b, 6.6b, and 6.8b. A reduced scope of candidates for merge operations showed to be a key factor on its performance.

DBMN displayed a lower number of evaluations per merge operations than S-DBMN, Fig. 6.5b, 6.7b, and 6.9b. However, its QoC yield improves significantly only after some cluster merges. This improvement in QoC yield is more evident after 7 cluster merges for the topologies with 9 level of connectivity, and 15 cluster merges for the topologies with 15 level of connectivity, Fig. 6.6b and 6.8b, respectively.

It is worth noting that for all the distributed algorithms there is no order of precedence

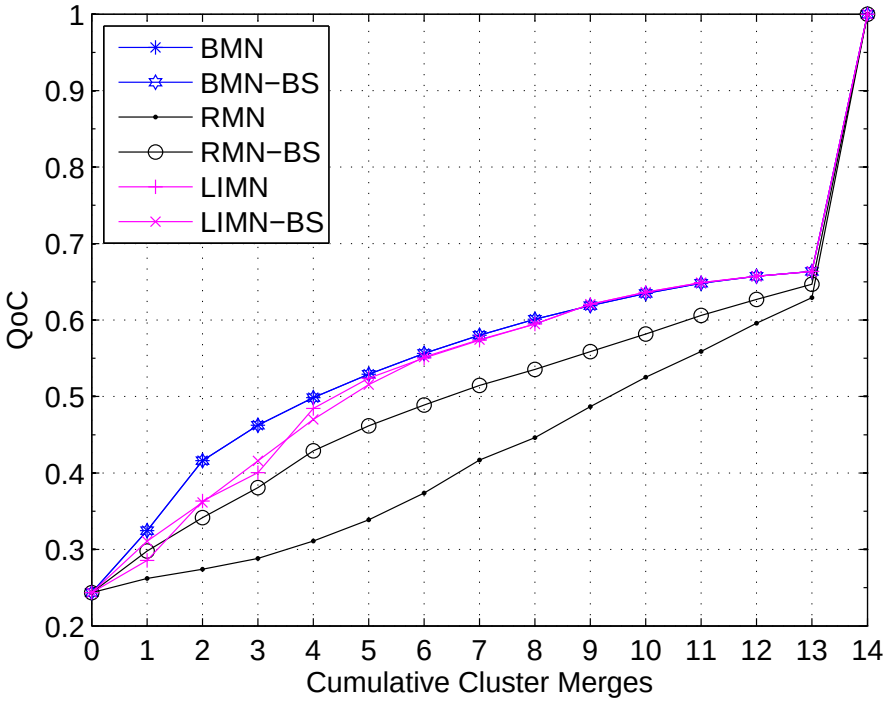
for merging operations to occur. For instance, if DBMN network merge operations order of precedence were to be sorted by QoC gain, then a QoC yield similar to that obtained by its centralized counterpart (BMN) would be observed. Such approach would have to be designed to be semi-distributed, and could benefit from a lower number of evaluations per operations than its centralized counterpart (BMN). Nevertheless, the communication overhead related to the cluster's synchronization for this approach has to be considered.

6.6.2.4 RMN, DRMN, and RMN-BS

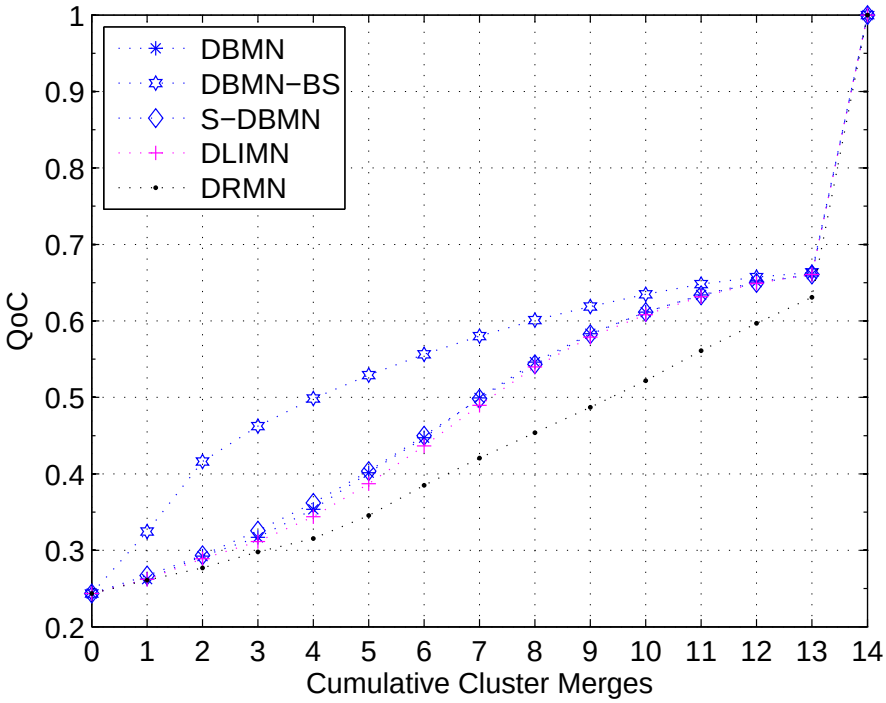
RMN and DRMN were selected as the base-case evaluations. All the presented algorithms show a QoC yield to be far superior to that obtained with RMN and DRMN algorithms.

In spite of its random nature, by significantly reducing the scope of candidates for merge operations to only neighbor clusters of the base station, RMN-BS shows an almost linear QoC yield. This is attributed to the repeated increase in the number of sensor nodes to the base station cluster. The base station cluster size has a higher weight factor in the QoC function due to the number of persistent communication links associated with it.

6.6.3 Performance Plots

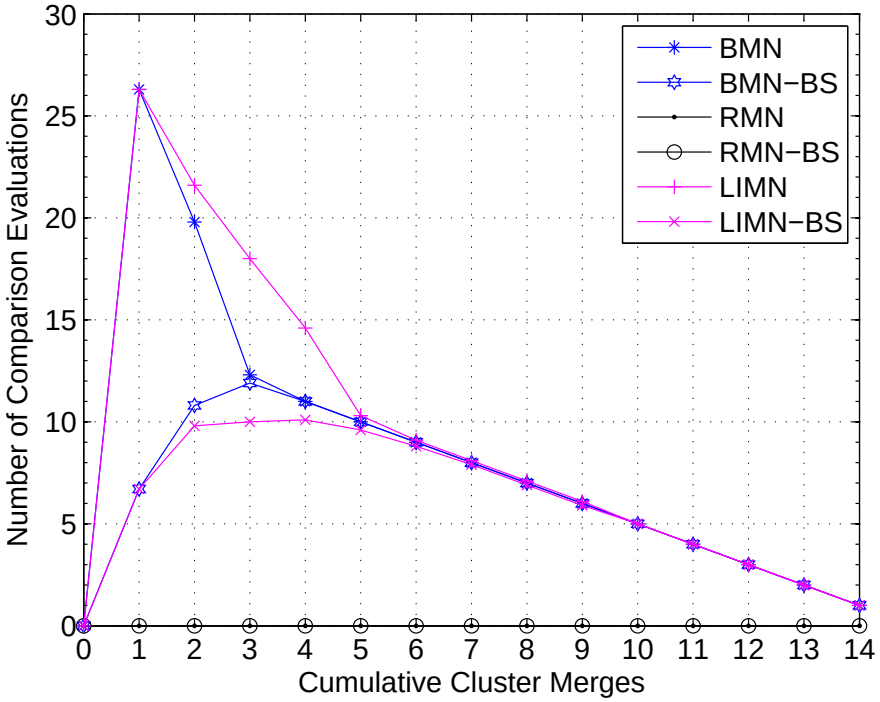


(a) QoC for Centralized Algorithms

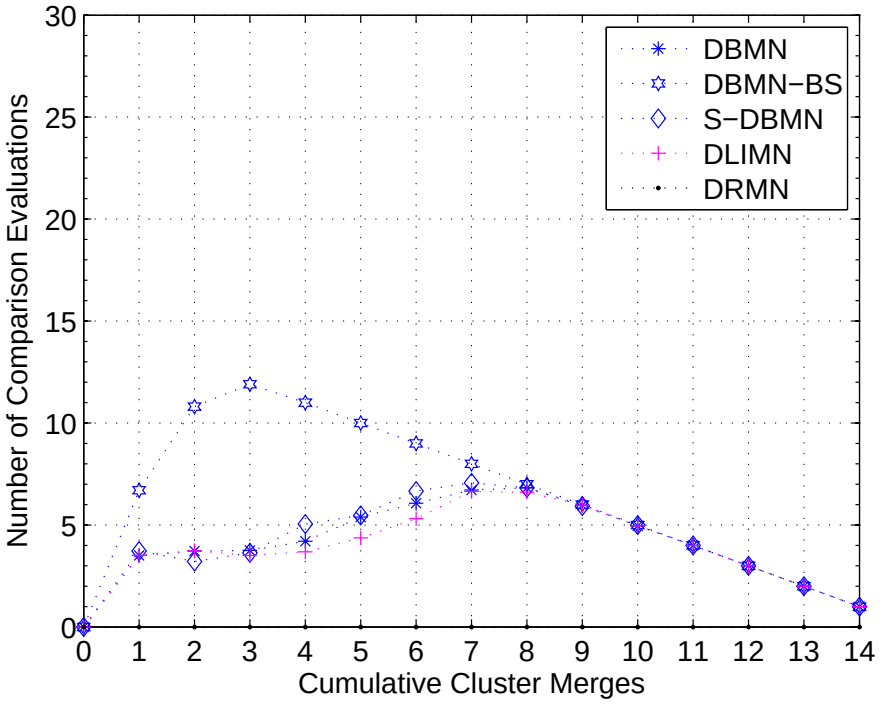


(b) QoC for Distributed Algorithms

Figure 6.4: QoC in Network Topologies with 3 Connectivity Levels, 15 Clusters

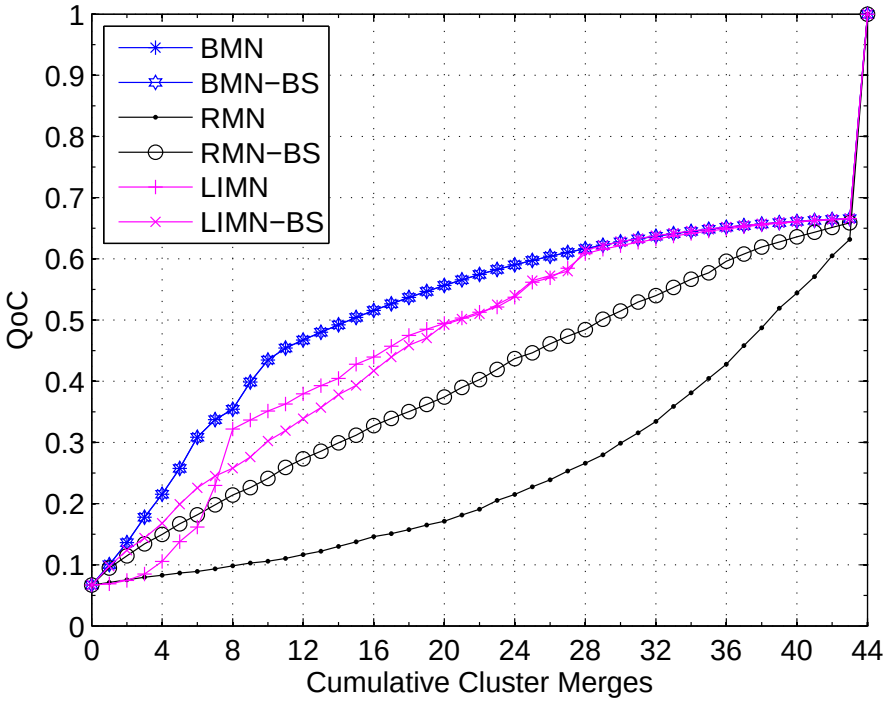


(a) Number of Evaluations for Cent. Algorithms

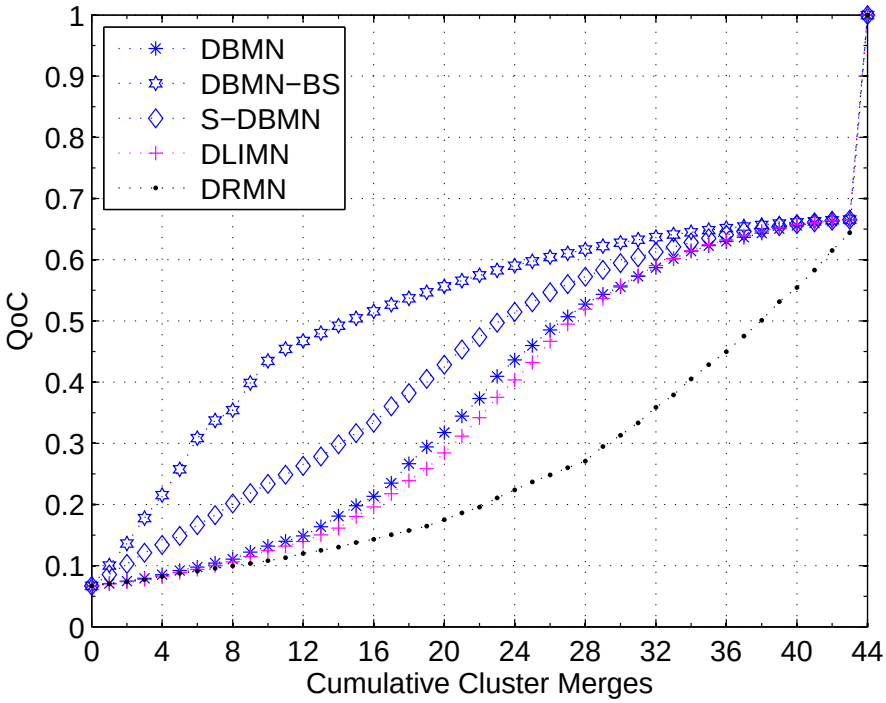


(b) Number of Evaluations for Dist. Algorithms

Figure 6.5: Number of Evaluations Per Merge Operation in Network Topologies with 3 Connectivity Levels, 15 Clusters

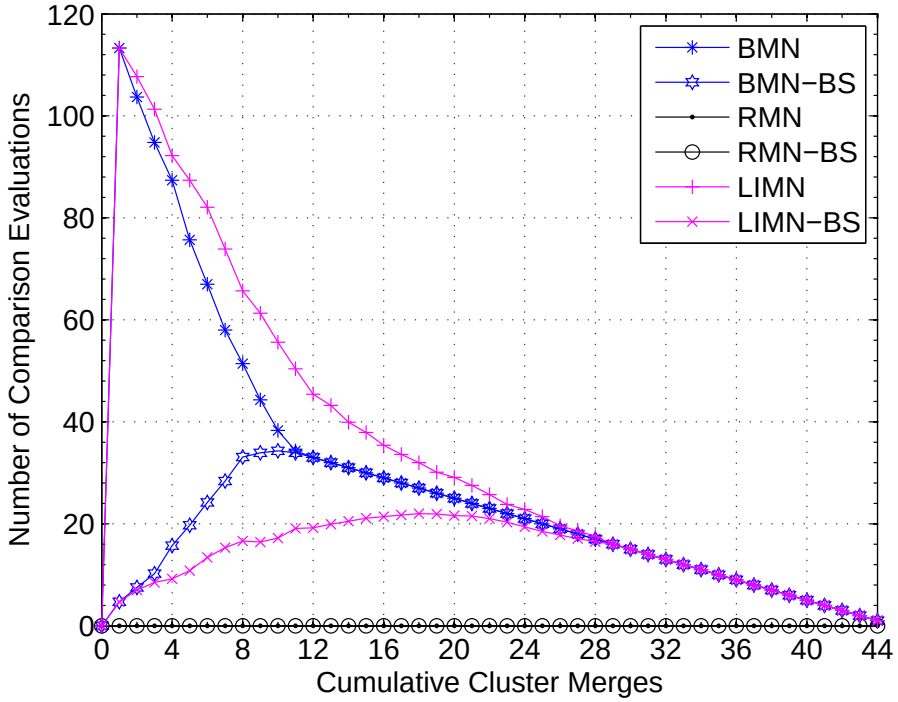


(a) QoC for Centralized Algorithms

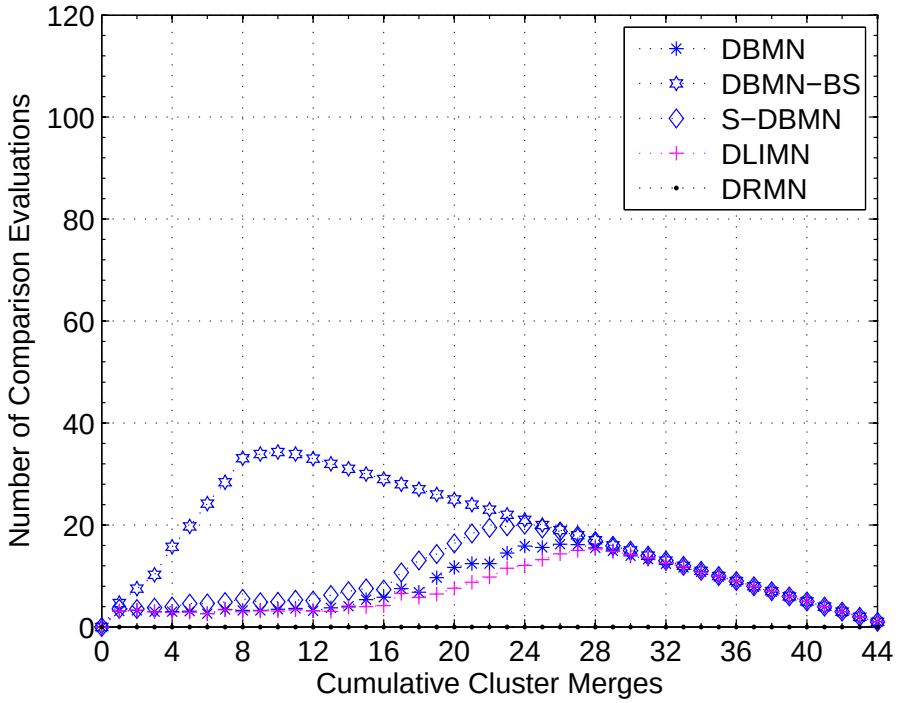


(b) QoC for Distributed Algorithms

Figure 6.6: QoC in Network Topologies with 9 Connectivity Levels, 45 Clusters

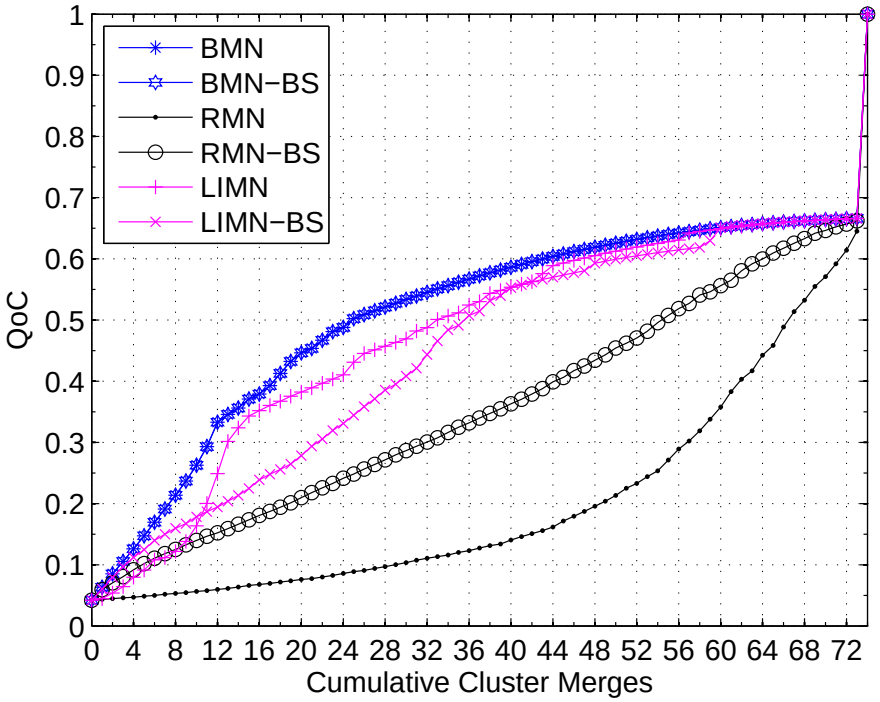


(a) Number of Evaluations for Cent. Algorithms

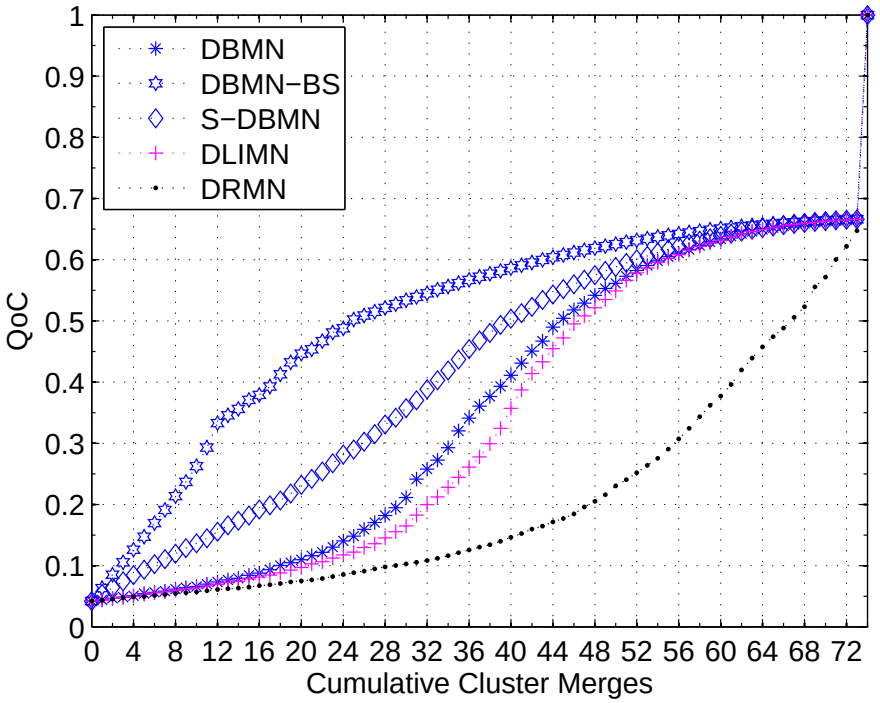


(b) Number of Evaluations for Dist. Algorithms

Figure 6.7: Number of Evaluations Per Merge Operation in Network Topologies with 9 Connectivity Levels, 45 Clusters

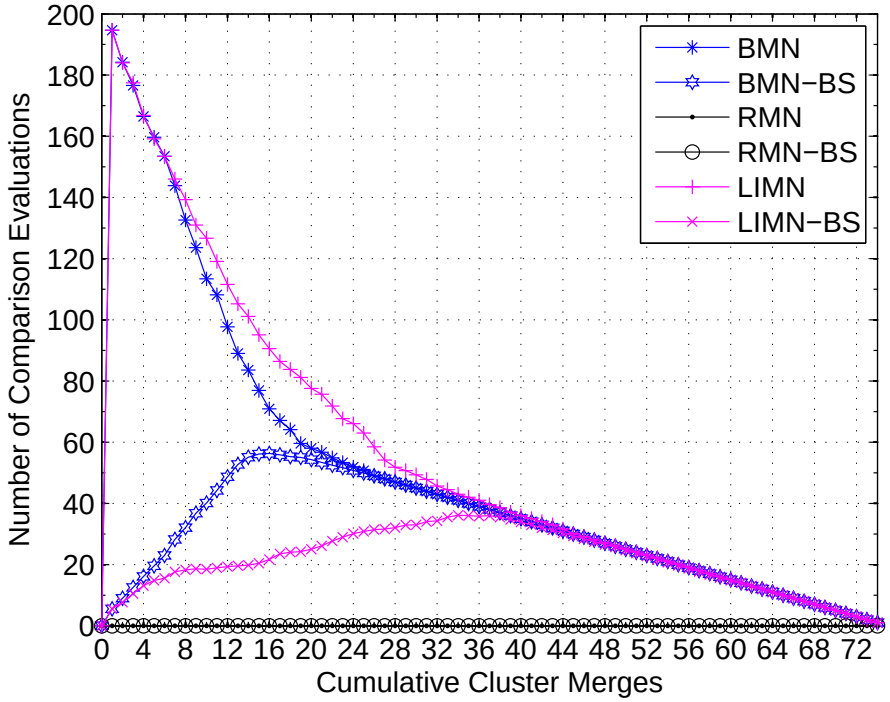


(a) QoC for Centralized Algorithms

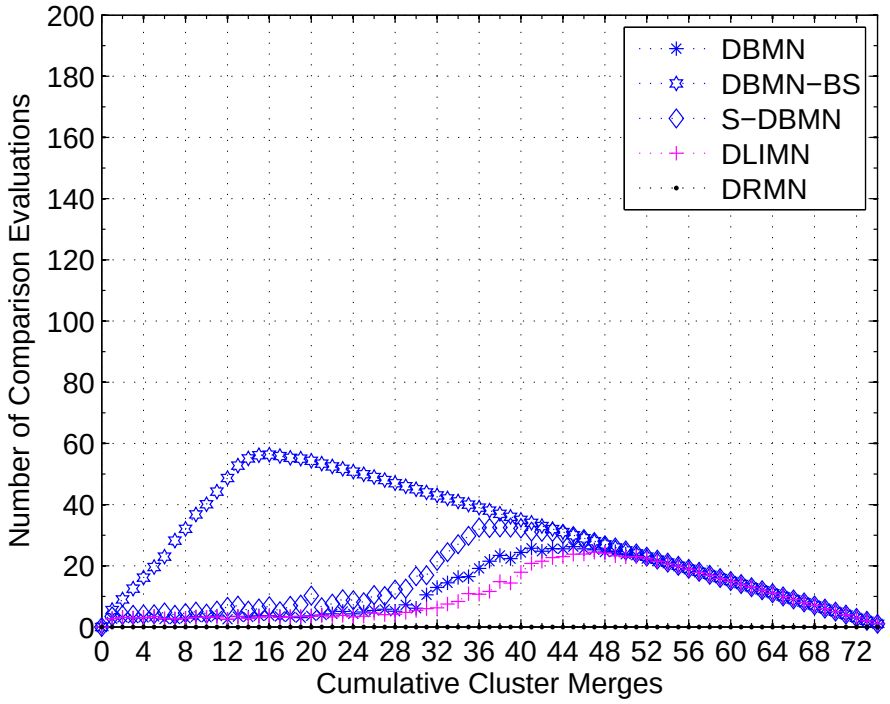


(b) QoC for Distributed Algorithms

Figure 6.8: QoC in Network Topologies with 15 Connectivity Levels, 75 Clusters



(a) Number of Evaluations for Cent. Algorithms



(b) Number of Evaluations for Dist. Algorithms

Figure 6.9: Number of Evaluations Per Merge Operation in Network Topologies with 15 Connectivity Levels, 75 Clusters

6.7 Summary

This chapter presents an unique fitness evaluation for wireless sensor network topologies. Network connectivity is crucial for a network topology to be functional. Nonetheless, network responsiveness and its capacity to meet communication flow demands is crucially important as well. This work evaluated network performance aspects related to network connectivity from the application's perspective. A Quality of Connectivity (QoC) performance metric was defined to evaluate the network connectivity in terms of how geographically distributed the sensor nodes are in the topology. Furthermore, the metric served as an indicator factor of potential communication problems and latency in sensor network topologies.

Centralized and distributed algorithm approaches are presented to maximize network performance. Maximizing network performance with fewer operations as possible is important for Wireless Sensor Networks, where energy and computing resources are limiting factors. Therefore, the algorithms make smart cluster merging operations attempting to yield maximum QoC for each operation. Network QoC improvement and the complexity of executing the approaches was evaluated in multiple and diverse network topology scenarios. From all the algorithm approaches, DBMN-BS was discussed to be less computing intensive than its centralized counterpart, BMN-BS, having an equally outstanding network QoC yield per operation.

As an extension of this work, it would be beneficial to design and evaluate a semi-distributed approach with maximum QoC yield decision criteria. Such approach will seek to combine the network QoC improvement performance obtained with DBMN-BS, and the low number of comparison evaluations per merge operations obtained with DBMN.

Chapter 7

LEVERAGING HEIGHT IN JUMPING SENSORS TO OBTAIN RELATIVE ORIENTATION

7.1 Preliminaries and Motivations

Jumping sensor robots have attracted attention for their unique ability to jump to target destinations, making them ideal candidates to be deployed in rough terrains where wheeled robots cannot perform, Cen and Mutka (2008); Pei *et al.* (2009). It is clear that as a newer generation of jumping sensors becomes available, opportunities open to solve many existing Wireless Sensor Network (WSN) problems, which include a node's relative orientation. We define a node's relative orientation as the ability of a node to identify the direction of a target (e.g., neighbor node) from its current location relative to a reference direction. Obtaining a reference direction, e.g., the magnetic north, has become simple with the usage of inexpensive magnetic compass sensors. However, the challenge remains to determine the relative direction of a neighbor node. While the orientation concept seems relatively simple in nature, it is

still a challenge for a highly resource constrained (CPU, power, form-factor, etc.) WSN.

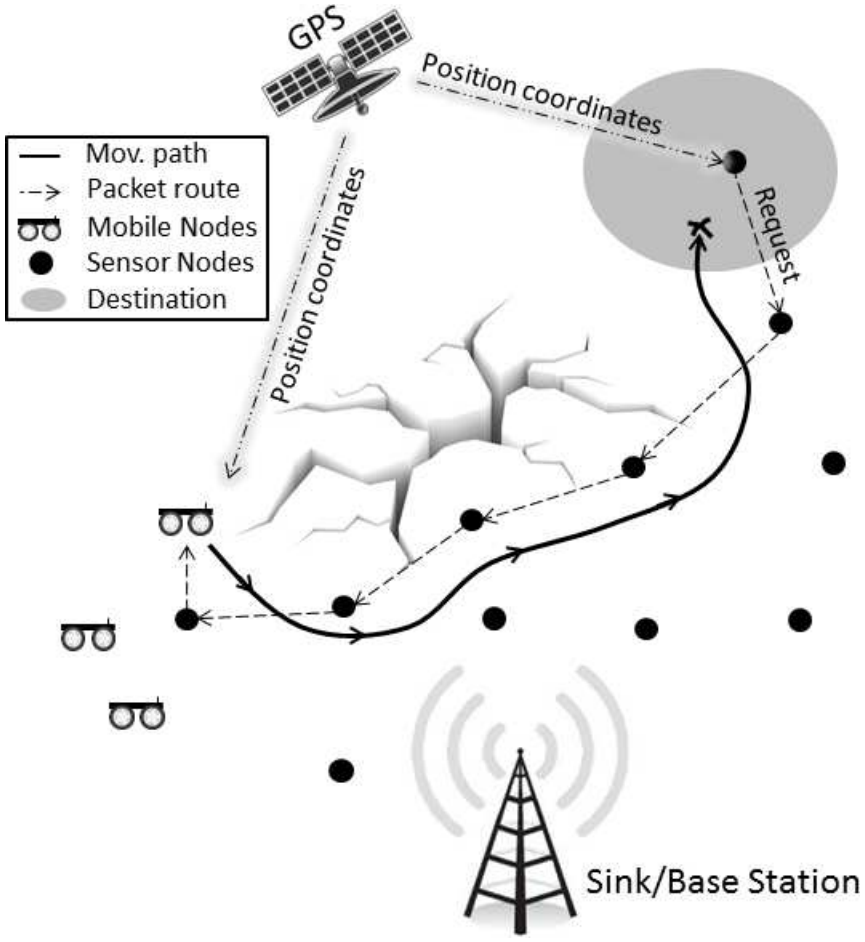


Figure 7.1: Orientation with GPS

Sensor orientation is a crucial area of importance for mobile wireless networks. Generally, node orientation is obtained from each node's location coordinates obtained by GPS receivers. Figure 7.1 illustrates a relocation request in a wireless network composed of a limited number of mobile nodes. When a node relocation request occurs, the relative orientation of the node to the target destination is computed from the position coordinates. The computation might take place in a base station if the node lacks sufficient computational resources. Nevertheless, localization is a requirement for those mobile networks to work. Enabling every device to be position coordinate aware can become a costly procedure and inconvenient for most power

limited mobile wireless sensor networks.

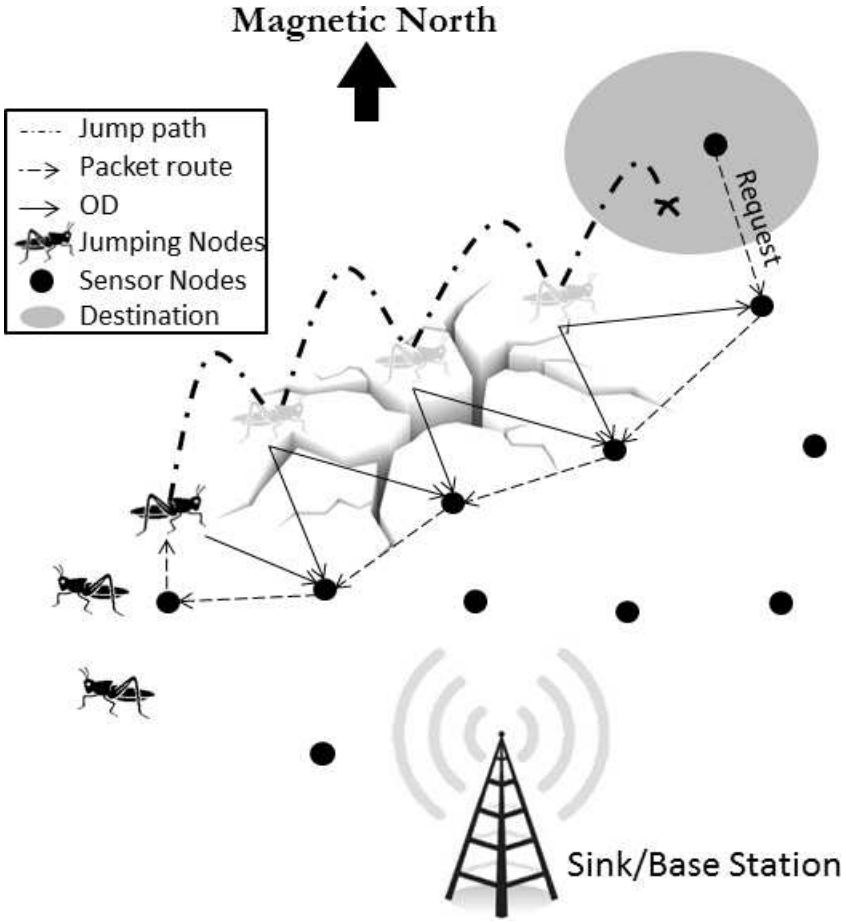


Figure 7.2: Relative Orientation with Jumping Sensors

The objective of this work is to leverage jumping and mid-air spinning to determine the relative orientation of sensors nodes without the initial need of position within a coordinate system. Even with the limitations of wireless sensors mentioned earlier, the fluctuations of the received signal strength as the antenna orientation changes provide sufficient information to predict relative orientation. The jumping ability allows the jumping sensor to relocate and while in this process, mid-air rotations can be used to scan the nearby nodes' relative orientation. The relative orientation data can be used to guide the relocation sensors as needed. For example, the destination of a mobile node can be defined as a progression of dynamic directions adjusted according to the relative orientation of surrounding nodes.

Figure 7.2 illustrates the movement progression of a jumping robot relying on the oriented directions (OD) from neighbor nodes. Every oriented direction is relative to the node itself and a reference direction. The magnetic north serves as a common fixed reference direction. Furthermore, with relative orientation, nodes can collaborate to identify network boundaries, uncovered areas, and the existence of obstacles in the topology. Sensor orientation is discussed further in Section 7.2.

This work presents a multi-step procedure to produce a direction oriented jumping sensor network. First, the jumping sensor collects and processes signal strength readings to produce the first prediction set of oriented directions per node. At this stage, the nodes have multiple possible directions for their neighbors. Second, this set is processed and reduced to two oriented directions, one being the true direction where the neighbor node lies, and the other its mirrored direction. Third, the process continues with the execution of a relative direction agreement algorithm, which selects and validates a single oriented direction per neighbor node. Finally, the direction oriented network executes a mirror free procedure, which detects and corrects mirrored oriented directions. Only a pair of beacon nodes is needed for the procedure to fix nodes' true orientation from a mirrored orientation.

7.1.1 Organization

This chapter is organized as follows. Section 7.2 introduces the jumping sensor approach for collecting and processing signal strength data into orientation information. A series of outdoor experiments are performed and the obtained results are evaluated. In Section 7.3 a directional-orientation decision algorithm is presented. Section 7.4 presents a mirror orientation identification and correction procedure. In Section 7.6 a summary of the work and conclusion are provided.

7.2 Sensor Orientation

In a three dimensional space, orientation of an object can be measured in terms of rotations around three axes: vertical (yaw), lateral (pitch), and longitudinal (roll). Figure 7.3 depicts a three dimension axes configuration where roll and pitch follow the right-handed rule for positive rotation, while yaw follows the left-handed rule positive rotation.

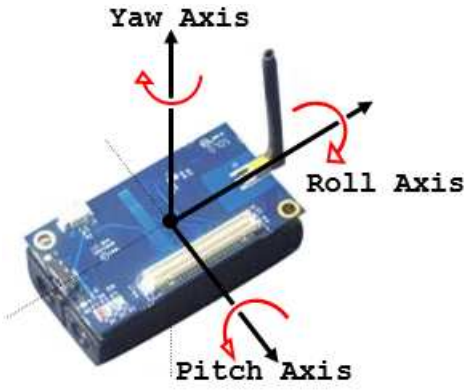


Figure 7.3: Yaw, Pitch, and Roll

The increase in demand for spacial orientation in technology gadgets, such as, smartphones, remote controllers, digital cameras, etc., have made orientation data acquisition sensors to be developed in small sized chips at an affordable customer price. Newest generations of wireless sensor robots are designed to have a magnetic compass, gyroscope, and accelerometer, all integrated as part of their control system. This makes orientation data available for robot repositioning and movement.

Most wireless sensors presently are equipped with dipole antennas. Dipole antennas radiate electromagnetic waves in a toroid manner, creating two radiation holes (where the radiation power drops significantly) along the direction of the antenna endings. Changes in the antenna orientation can make nearby receivers to register higher or lower signal strength, depending where the radiation pattern of the antenna is concentrated relative to the receivers.

If the antenna orientation is changed intentionally, lower signal strength readings from nearby receivers can potentially indicate where the radiation holes are, and relative direction from receivers could be identified.

Jumping sensors are capable of two-way communication with a far receiver while they are airborne, for relocation or simply for node discovery, as presented in Cintrón and Mutka (2010). While the jumping sensor is airborne it freely rotates on the air, changing the orientation of its antenna. Mid air rotations are beneficial because it allows the sensor to transmit at different antenna orientations. Receiver of airborne communication packets can process the RSSI of received packets to identify drops in signal power.

This section will present results from a series of experiment conducted where antenna orientation was manipulated to create fluctuations in the RSSI registered in a receiver node. The processing of this data makes it possible to identify a node's relative orientations.

7.2.1 Oriented Direction

In wireless sensors, sensor nodes within radio range of each other are called neighbors. We define the direction of a node's neighbor with respect to a common reference as the oriented direction (OD) of the neighbor from the node. The reference can be the earth magnetic north pole, a fixed land mark or object with which the sensors can get oriented. Once a common reference is established, the OD of a neighbor is the vertical axis rotation angle (yaw) that points in the direction toward the neighbor. Hence, it is worth noting that ODs are relative to the observer node.

When a jumping sensor jumps, it can broadcast its orientation in communication packets during the entire time of flight. Neighbor receivers register the RSSI for each broadcast packet. The RSSI is associated to the orientation of the transmitter (at transmission time)

that is enclosed in the packet. The OD of a receiver node is obtained by finding the orientation with minimum RSSI.

Due to the nature of dipole antennas, two coplanar 180 ° apart directions have to be taken into consideration when identifying the direction of the neighbor. Both directions, one identified with minimum RSSI and the other being the 180 °-complement, are assigned equal probability to be the true direction.

7.2.2 Experimentation

A series of outdoor experiments were performed to obtain Oriented Directions of two wireless sensor nodes. Two IRIS[1] motes where used in the experiments. Maximum transmission power (3 dBm) was employed for wireless communication. The sensors were separated at a 5 meter distance on a flat surface. The orientation of the transmitter was changed as follows. Eight (8) Rotations on a vertical axis (yaw) were set at increments of 45 degrees, for a complete 360 degree rotation. The antenna in the IRIS is fixed to what we define as the front side (for experimentation purpose), with a single axis rotation connector, allowing antenna rotations along the longitudinal axis (roll). Four (4) roll angles were selected over 180 ° above the horizontal plane: $-45\,^\circ$, $-90\,^\circ$, $45\,^\circ$, $90\,^\circ$. It is worth noting that for any roll angle, the antenna will be always perpendicular to the longitudinal axis.

Each yaw orientation was combined with all four roll angles to form a total of 32 antenna orientations, i.e., 8 yaw by 4 roll angles. Two fixed heights were tested: 0 cm and 50 cm above the surface. Only the height of the transmitting sensor was changed, leaving the receiver at surface level at all times. All the antenna orientations were tested for each height, repeating the process for the second node set as the transmitter. Fifty (50) sets of ten (10) packets

[1] `http://www.memsic.com/wireless-sensor-networks/`

were transmitted for each antenna orientation. The receiver node was set to register the RSSI for each packet, and the mean RSSI was computed for each orientation.

In order to obtain the ODs for each node we need to find the antenna orientations having the minimum RSSI. However, we are only interested in the yaw angle of the antenna itself, not the sensor. Therefore, it is required to obtain the direction component, from the antenna orientation, in the horizontal plane of reference. Note that this will not be required if the antenna cannot change orientation independently from the sensor. In our experiment, the sensor roll was always kept fixed, and the antenna roll adjusted to the desired angle.

To obtain the antenna yaw angle (in our experiment) we needed to add $\pm 90^\circ$ (the sign is determined by the sign of the roll angle) to the sensor yaw orientation. This is because the horizontal plane of reference is perpendicular to the vertical axis. Table 7.1 presents the true antenna yaw orientation for each combination of sensor's yaw orientation and antenna roll. For example, if the sensor yaw is 90°, the antenna yaw angle will be 0° for a negative roll, and 180° for a positive roll. Similarly, a sensor yaw 180°-complement will result in the same antenna yaw angles but in opposite direction with respect to the roll sign. Therefore, we end up with four RSSI measurement corresponding to every yaw angle, as a result from the translation of the sensor yaw readings to the antenna perspective.

Table 7.1: Antenna True Yaw Orientation for IRIS mote

	Sensor Yaw							
Roll	0	45	90	135	180	225	270	315
-90,-45	270	315	0	45	90	135	180	225
45, 90	90	135	180	225	270	315	0	45

The RSSI mean was computed for the small samples of ten (10) packets for each antenna yaw and roll configuration. RSSI results are shown by height for node A at different yaw angles while transmitting at -45° and -90° antenna roll, Fig. 7.4 and Fig. 7.5, respectively.

RSSI results for node B followed a similar pattern. It should be noted that for experimental purpose, each node reference (0 ° yaw) was fixed to point to toward each other. From Fig. 7.4, it can be appreciated that node A's lower RSSI were registered at 0 ° and 315 ° yaw for a transmitting height of 0 cm, and 0 ° yaw for 50 cm height. RSSI readings for −90 ° roll (Fig. 7.5) with low values were registered at 0 °, 270 °, and 315 ° yaw angles, for 0 cm height. Although 315 ° yaw RSSI readings had the lowest value when at 50 cm height, both 0 ° and 180 ° yaw produced the second lowest RSSI.

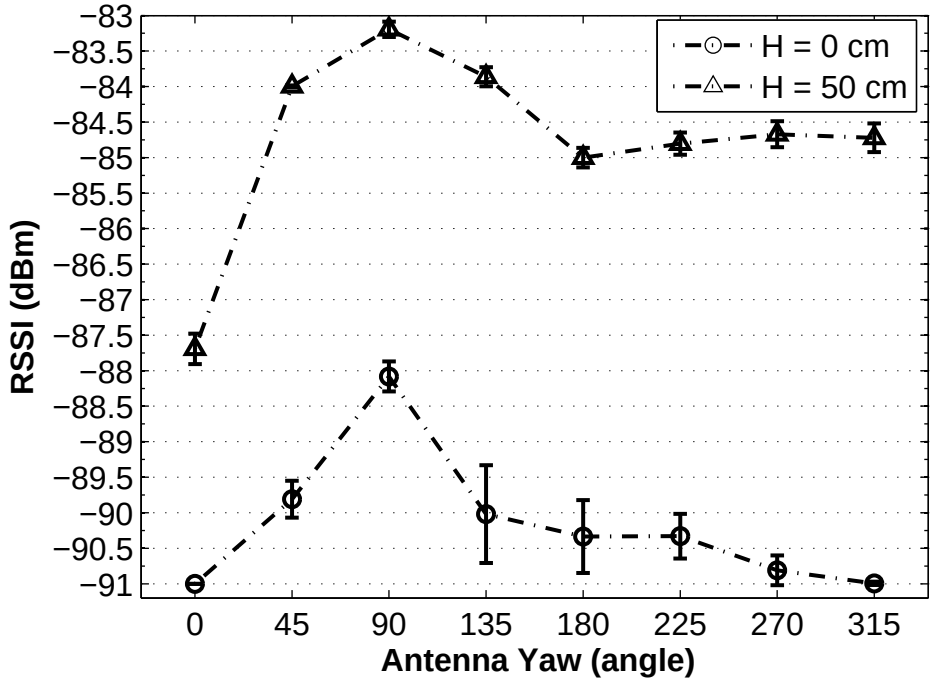


Figure 7.4: Node A's Antenna Yaw RSSI (w/ Roll −45°) Measured at Node B

The RSSI readings showed a high correlation with the antenna orientation. Some degree of offset to the expected yaw angle direction was experienced. This was due to a number of factors. First, the shape of the antenna conductor in the motes is not perfectly straight, which can create changes in the radiation pattern. Second, the antenna location on the motes is at its side, not at its center, making it almost half visible from the opposite side, which can interfere with the signal. A rigid antenna conductor strategically positioned on

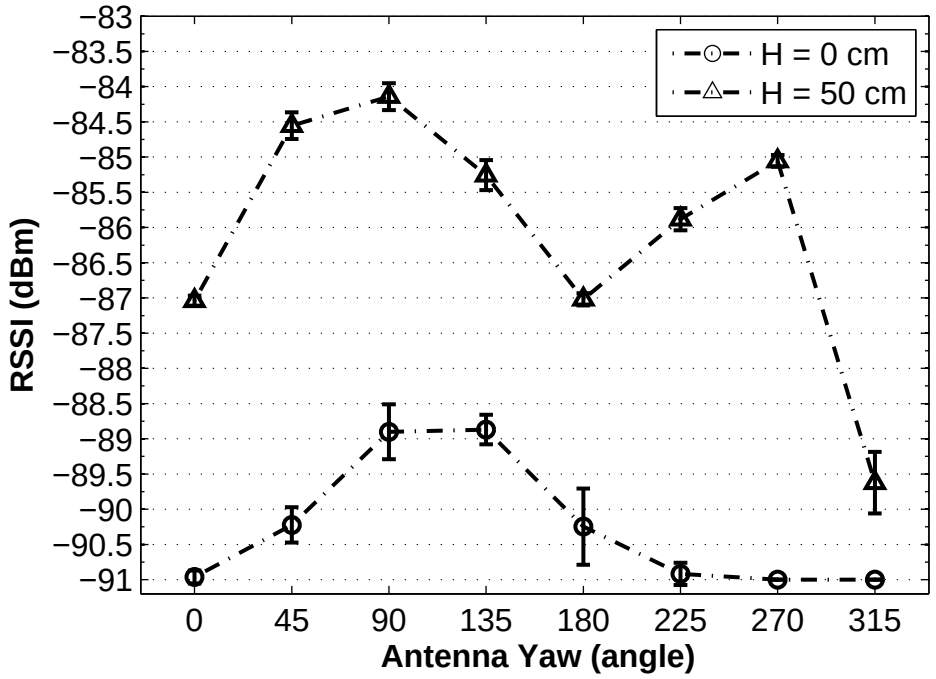


Figure 7.5: Node A's Antenna Yaw RSSI (w/ Roll $-90°$) Measured at Node B

the robot's body can mitigate the influence of these. Third and foremost, when the mote is at surface level much of the signal is absorbed by the surface, in addition to any line-of-sight obstruction between the transmitter and receivers, e.g., unevenness of the surface. The ability of a jumping sensor to perform surface and airborne communication makes it adequate for OD detection. Furthermore, while the mote rotates, transmission can occur at smaller yaw intervals, providing a higher RSSI resolution per rotation.

7.2.3 Multiple Minima

Multiple RSSI minima can be found at different yaw angles. A solution to this problem in its simpler form can be obtained by computing the average of yaw angles corresponding to the minimum RSSI values. However, the yaw angles of the minimum RSSI values may not always lie close to each other due to the dipole antenna effect. In such case, the average

value alone will most likely result to an offset direction far from the true OD. To deal with this we propose an OD estimator (Alg. 7.1).

Algorithm 7.1 Estimate OD from a set with n directions

1: $AOS_{all} \leftarrow$ Angle of separation from all pair of directions.
2: **if** $n = 2$ **then**
3: **if** $AOS_{1,2} \leq 120$ **then**
4: $OD_1 \leftarrow$ mean([yaw $\angle_1$, yaw $\angle_2$])
5: **else**
6: $OD_{1a} \leftarrow$ yaw $\angle_1$
7: $OD_{temp} \leftarrow$ yaw $\angle_2$
8: $OD_{1b} \leftarrow$ 180complement(OD_{temp})
9: $OD_1 \leftarrow$ mean($[OD_{1a}, OD_{1b}]$)
10: **end if**
11: **else**
12: $NearestPair \leftarrow$ Find nearest pair in AOS_{all}
13: $OD_1 \leftarrow$ mean($NearestPair$)
14: $CurrentSize \leftarrow 3$
15: $PreviousSize \leftarrow 0$
16: **while** $PreviousSize \neq CurrentSize$ **do**
17: $NewSet \leftarrow$ All directions within $\pm 60\,^\circ$ from OD_1
18: $OD_1 \leftarrow$ mean($NewSet$)
19: $PreviousSize \leftarrow CurrentSize$
20: $CurrentSize \leftarrow |NewSet|$
21: **end while**
22: **end if**
23: $OD_2 \leftarrow$ 180comp(OD_1)

The OD estimator computes the mean direction from the multiple yaw minima within $120\,^\circ$ yaw range. The yaw range gives enough margin ($\pm 60\,^\circ$) to include offset readings from the true direction, while at the same time avoids including $180\,^\circ$-complement readings created by the dipole antenna effect. If there are only two minima ($n = 2$) with a separation angle less or equal $120\,^\circ$, the mean direction is computed (Alg. 7.1, line 4), else the mean of one direction (randomly chosen) and the $180\,^\circ$-complement of the other is assigned as OD_1 (Alg. 7.1, lines 6-9). For cases with $n \geq 3$ the algorithm starts by computing the mean direction (OD_1) from the closest (smallest angle of separation) minima pair (Alg. 7.1,

lines 12-13). Then, OD_1 is recomputed from the mean of all minima within $\pm 60^\circ$ from the previous OD_1 (Alg. 7.1, lines 17-20). This process continues until there is no change in the minima enclosed in the 120° range of the computed OD_1. Once OD_1 is set, the 180°-complement OD_2 is computed (Alg. 7.1, line 22).

The OD estimator algorithm reduces multiple ODs with minimum RSSI to only two ODs, the true OD and its mirror. A network-wide OD algorithm to reduce the ODs of each node to one per neighbor is presented in Section 7.3.

7.3 Orientation Algorithm

This section presents the process to reduce the possible ODs per neighbor sensor node to a single OD. It is assumed that each node will have two ODs that are 180° apart for each neighbor, both having equal probability of being the true OD. Therefore, it is important that the node's OD choice is kept consistent and valid with its neighbors.

To illustrate the problem, let's analyze the ODs for 3 nodes as shown in Fig. 7.6. Each node has two OD per neighbor. One is the true OD (OD_{true}) where the neighbor really lays, and the other 180°-complement is what we will refer as the mirror OD (OD_{mirror}). If node A chooses node B's OD_{true}, then to keep consistency, node B must choose node A's OD_{true} from its perspective. The same applies for mirror ODs. Generally, if node B's OD from A is OD_{AB}, then node A's OD from B must be the 180°-complement of OD_{AB}, as expressed in Equation (7.1).

$$OD_{BA} = 180comp(OD_{AB}) \tag{7.1}$$

Every nodes' OD choices must be validated, i.e., comply with an established rule or a set of rules. If a set of three non collinear nodes are neighbors, then a triangle can be

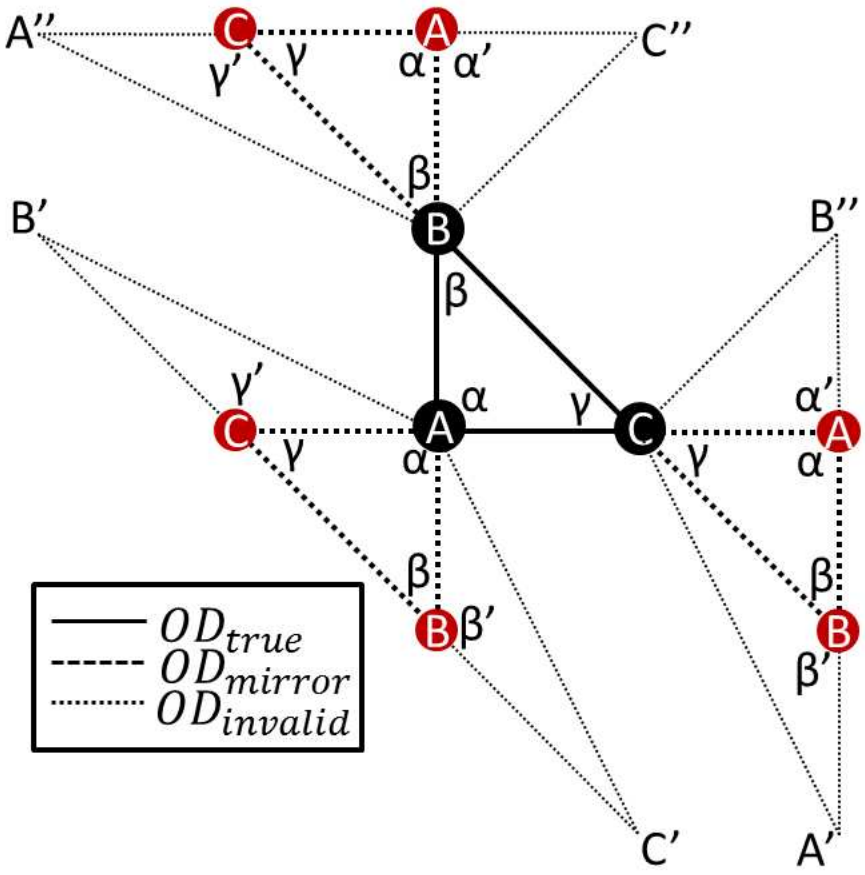


Figure 7.6: ODs Choice Problem

established with nodes as vertices, and lines between directly connected nodes as edges, as shown in Fig. 7.6. Since each edge is associated with an OD, interior angles can be obtained. In order for the ODs for three nodes to be valid they must meet the triangle's sum of interior angles (180 °) rule of Equation (7.2). Generally, any triangle formed by three connected and non collinear nodes labeled A, B, and C, have interior angles $\angle BAC$, $\angle ABC$, and $\angle BCA$, that can be represented as α, β, and γ, respectively.

$$\alpha + \beta + \gamma = 180 \tag{7.2}$$

From Fig. 7.6, it can be appreciated that mirror ODs comply also with Equation 7.2. This

is possible when the OD choice is kept consistent, i.e., either the ODs are all true ODs or all mirror ODs. If all ODs in the network are mirrors, then we call it a mirror oriented network. A mirror oriented network can be transformed into the true orientation by replacing all ODs with their 180 °-complement, which is discussed in Section 7.4. The remaining of this section discusses the OD selection process.

7.3.1 OD Selection

The OD selection process is presented in Algorithm 7.2. The objective of the algorithm is to assign valid ODs that keep consistency with those previously selected. A node is selected as an anchor node (A) to initiate the selection process (Alg. 7.2, line 1). The anchor node can be selected by quorum agreement, e.g., if unique IDs are used to identify nodes, then a node with the minimum or maximum ID can be selected. Node A's neighbors with OD are kept in $Oriented_A$, while those needing a decision are kept in $NonOriented_A$. Collinear nodes are kept in *Collinear* until a valid OD can be assigned to them. It is likely that a node's neighbors are within communication range of each other. We call the common neighbors (*Commons*), line 7. If there are neighbors with assigned OD, then the algorithm proceeds to choose ODs for those other neighbors that are common to them, but not oriented (*Comnor*), line 8.

Once an oriented node is selected (B), the ODs of nodes A and B are divided in two zones (lines 13-14), using OD_{AB} and OD_{BA} as a zone divider. It is worth recalling that OD_{BA} is the 180 °-complement of OD_{AB}. A third node (C) is selected from those that are common with B but still not oriented (line 15). The anchor node (A) makes the first attempt to assign an OD to C (OD_{AC}), (line 16). This OD selection is a random choice between both of the ODs for node C, $OD1_{AC}, OD2_{AC}$. Next, the anchor selects the OD of

node C, from the perspective of node B (OD_{BC}), that lays in the same zone as (OD_{AC}). Validation is done using Eqn. 7.2 with the obtained angles of separation from each pair of

Algorithm 7.2 Neighbor OD Selector

```
1: A := Anchor node
2: Oriented_A ← A's neighbors with OD
3: NonOriented_A ← A's neighbors without OD
4: Collinears ← ∅
5: repeat
6:     for all B ∈ Oriented_A do
7:         Commons ← NeighborsOf_A ⋂ NeighborsOf_B
8:         Comnor ← Commons ⋂ NonOriented_A
9:         Zone1_A ← A's ODs laying within OD_AB + 179°
10:        Zone2_A ← A's ODs laying within OD_AB − 179°
11:        Zone1_B ← B's ODs laying within OD_BA + 179°
12:        Zone2_B ← B's ODs laying within OD_BA − 179°
13:        Z_1 ← Zone1_A ⋃ Zone2_B
14:        Z_2 ← Zone2_A ⋃ Zone1_B
15:        for all C ∈ Comnor do
16:            OD_AC ← OD1_AC
17:            if OD_AC ∈ Z_1 then
18:                if OD1_BC ∈ Z_1 then
19:                    OD_BC ← OD1_BC
20:                else
21:                    OD_BC ← OD2_BC
22:                end if
23:            else                                          ▷ OD_AC ∈ Z_2
24:                if OD1_BC ∈ Z_2 then
25:                    OD_BC ← OD1_BC
26:                else
27:                    OD_BC ← OD2_BC
28:                end if
29:            end if
30:            Tries = 2
31:            repeat
32:                if α + β + γ ≠ 180 then                   ▷ Validation
33:                    if Tries > 1 then
34:                        OD_AC ← OD2_AC
35:                        OD_CB ← 180comp(OD_CB)
36:                    else                                  ▷ Collinear nodes!
37:                        OD_AC ← ∅
38:                        OD_CB ← ∅
39:                        Include C in Collinears
40:                    end if
41:                else                                      ▷ Valid; must keep Consistency
```

Algorithm 7.2 Neighbor OD Selector (cont'd)

```
42:                 OD_CA ← 180comp(OD_AC)
43:                 OD_CB ← 180comp(OD_BC)
44:                 Append C to Oriented_A
45:                 Delete C from NonOriented_A
46:                 break                                   ▷ Exit Tries loop
47:             end if
48:             Tries ← Tries − 1
49:         until Tries = 0
50:     end for                                             ▷ end for all C ∈ Comnor
51: end for                                                 ▷ end for all B ∈ Oriented_A
52: Collinears ← NonOriented_A ∩ Collinears
53: NonOriented_A ← NonOriented_A − Collinears
54: LeftCount ← |NonOriented_A|
55: if LeftCount > 0 then                                   ▷ Predictive Assignment
56:     DONE ← FALSE
57:     ODprediction(A, Oriented_A, NonOriented_A)
58:     NewLeftCount ← #NonOriented_A
59:     if LeftCount − NewLeftCount = 0 then
60:         DONE ← TRUE                                     ▷ No new OD
61:     end if
62:     NonOriented_A ← NonOriented_A ∪ Collinears
63: else
64:     DONE ← TRUE
65: end if
66: until DONE
```

OD per node. If the ODs selections are not valid, the 180 °-complements are selected and re-evaluated (lines 32-35). If a second re-evaluation fails validation, the node is considered collinear (line 39). Otherwise, when OD selections are found to be valid, consistency is kept by assigning the 180 °-complements of OD_{AC} and OD_{BC} to OD_{CA} and OD_{CB}, respectively (lines 42-43).

There are two circumstances where an OD prediction is needed. First, OD prediction is needed when a node does not have oriented neighbors. Second, it is needed when the anchor node and remaining neighbors without ODs do not share common oriented neighbors. Algorithm 7.2 identifies these cases by keeping track of nodes left without OD (lines 54-65). At this stage in the algorithm, if there are nodes without OD, and not identified to be

collinear, an OD prediction is needed before running a new iteration of the OD selection.

Algorithm 7.3 OD Prediction

```
 1: A := Anchor node
 2: Oriented_A := A's neighbors with OD
 3: NonOriented_A := A's neighbors without OD
 4: ODfound ← FALSE
 5: OrientedCount ← #Oriented_A
 6: if OrientedCount > 0 then                                  ▷ Use Oriented nodes
 7:     Clusters_A ← Sort Oriented_A into Clusters
 8:     i = 1
 9:     for all cluster ∈ Clusters_A do
10:         {bound_1, bound_2} ← GetBoundaries(cluster)
11:         NoZone[i] ← {bound_1 − 60°, bound_2 + 60°}
12:         i = i + 1
13:     end for
14:     for all B ∈ NonOriented_A do
15:         for i = 1 → #Clusters_A do
16:             if OD1_AB ∈ NoZone[i] then
17:                 OD_AB ← OD2_AB
18:                 ODfound ← TRUE
19:                 GoTo FinishLabel (line 32)
20:             else if OD2_AB ∈ NoZone[i] then
21:                 OD_AB ← OD1_AB
22:                 ODfound ← TRUE
23:                 GoTo FinishLabel (line 32)
24:             end if
25:         end for
26:     end for
27: end if
28: if ¬(ODfound) then
29:     B ← Select NonOriented_A having max #Commons
30:     OD_AB ← Random choice {OD1_AB, OD2_AB}
31: end if
32: FinishLabel:
33: OD_BA ← 180comp(OD_AB)                                     ▷ Consistency
34: return
```

OD prediction is handled in the *ODprediction* function, described in Algorithm 7.3. The function first checks for existing anchor's oriented neighbors. If there are none, the node with more common neighbors to the anchor is selected and the anchor's OD to the node (OD_{AB}) is selected randomly from the two possible ODs (Alg. 7.3, lines 28-31). Finally,

the 180 °-complement of OD_{AB} is assigned to OD_{BA} (Alg. 7.3, line 33).

When there are oriented neighbors, a prediction is done by avoiding conflicting zones. A conflict zone (NoZone) is the region where the OD of an oriented neighbor (not common neighbor with the non-oriented nodes) lays. The size of the NoZone is established with two boundaries, $\pm 60\,^{\circ}$ to the OD. Similarly, for regions with multiple neighbors (cluster), the pair of ODs having the greatest angle of separation is used, and $\pm 60\,^{\circ}$ is added respectively to establish the two boundaries of the NoZone cluster (Alg. 7.3, lines 10-11). After having established all the NoZones, both potential ODs from each $NonOriented_A$ neighbor node are checked to see if any lays in a NoZone. A potential OD found to be in a NoZone is discarded and its 180 °-complement is used as the anchor's OD for that node (OD_{AB}), (Alg. 7.3, lines 15-25). Likewise, the 180 °-complement is assigned as the node's OD to anchor (OD_{BA}) (Alg. 7.3, line 33). Since a prediction can have an impact on other ODs, the *ODprediction* function culminates after one OD is predicted, thus resulting in a new oriented node.

The new oriented node is used in Alg. 7.2 to start a new iteration. Also, collinear nodes have the opportunity to be oriented in further iterations, as more OD are established. Finally, the algorithm concludes either, when there are no more neighbors left without orientation (Alg. 7.2, lines 63-64), or when the available information to the anchor is not enough to predict neighbors' ODs (Alg. 7.2, lines 59-61).

7.4 Mirror Free Topology

In Section 7.3, an algorithm to choose valid ODs while keeping consistency among the network was presented. However, the resulting topology could be mirrored. In a mirrored topology, ODs from nodes are pointing 180 ° away from the true ODs. This topology still

holds validity and consistency, but a transformation is needed to make it usable by sensor nodes. In this section, we will discuss an innovative technique to detect and rearrange mirrored ODs in an oriented network topology.

We have departed from the notion that the information available to the nodes is not enough to discriminate between a true and a mirror OD. Therefore, the topology needs extra information to detect mirrored ODs. Nodes need to have known references in the field, such as beacons, in order to distinguish true ODs from mirror ODs. However, precise deployment of references can become difficult in many scenarios, and will increase the overall cost of deployment and maintenance of the network. Therefore, the usage of beacons should be limited.

Normally, a set of three to four non collinear beacons are used to triangulate the relative position of an object. This position can be transformed to a global coordinate system when these beacons posses geographic coordinates. However, our topology already has relative orientation information, which can be used by nodes. Distance estimation between beacons and nodes is not needed. The orientation information can be used in conjunction with a reduced set of oriented beacon nodes to identify mirrored ODs.

7.4.1 OD Paths and Mirrored Topology Identification

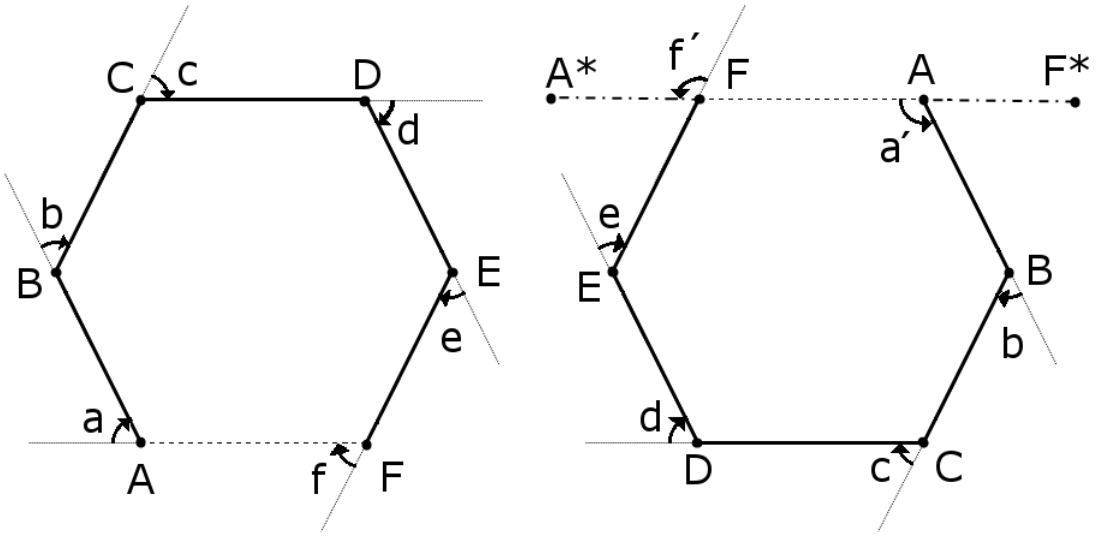
Routing paths between nodes in a network topology with ODs can be used to keep track of the change in direction, relative to each node in the path, as the path is traversed by a packet. We define these paths as OD-paths. Each node from the set belonging to an OD-path is required to have the OD of the previous (incoming) and next (forwarding) node in path.

Let us consider the nodes in an OD-path to be vertices and the ODs between them be

the edges of a graph. If we add an additional edge between source node and destination node in path, we form a closed loop, which is also a polygon. Polygons share an interesting characteristic to our interest. It is known that the sum of the exterior angles, with one adjacent side, of a polygon results in 360 °. Therefore, the sum of the changes in directions in an closed-loop OD-path equals to 360 °.

Now if we can guarantee that the additional edge between source and destination will always have true ODs, then a mirrored OD-path can be identified as follows. Adding all the changes in direction from a true OD path will result in 360 °. However, when the OD-path is composed of mirrored ODs, but not the OD's between source and destination, then the sum of changes in direction is not 360 °.

A pair of beacon nodes having their location information (e.g. global coordinates) can compute each other's true OD to create the closed-loop OD-path. Fig. 7.7a illustrates a true OD-path with two beacons at the end points, A and F. The mirror identification process starts with beacon A sending a communication packet to beacon F, including its location



(a) True OD-path (b) Mirrored OD-path

Figure 7.7: OD-paths

information and the OD of the first node in the path (OD_{AB}). Also, a total change in direction register (T) is kept, and updated as the packet traverses the path. Each node in the path is responsible for updating the change in direction before the packet is forwarded. Once the packet arrives to beacon F, the location information of A is used to obtain its true oriented direction from F, OD_{FA}. Beacon F updates the change in direction register with the two missing direction changes. (1) the change in direction f, as if the packet were to be forwarded directly to beacon A. (2) the initial change in direction a, i.e., change from OD_{FA} to OD_{AB}. The path is identified as true after verifying the total direction change, $T = 360\,^{\circ}$.

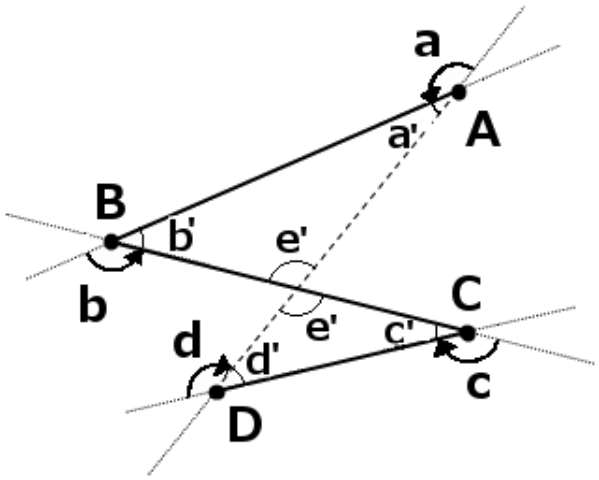
Similarly, Fig. 7.7b illustrates the mirrored OD-path. From the figure, we can see that even if the path is mirrored, the true ODs from beacons are used, i.e., OD_{FA*} and OD_{AF*}. This is due to the fact that the beacons can obtain their absolute position. Therefore, the changes in direction considered from $F*$ to A, and from $A*$ to F are the supplementary angles a', i.e., $a + a' = 180\,^{\circ}$, and f', i.e., $f + f' = 180\,^{\circ}$. When adding all the changes in directions, we have $T = 0\,^{\circ}$.

Since the oriented topology is kept consistent (see Section 7.3), a true OD-path indicates that the nodes covered by the beacons are true oriented. It should be noted that a node can be covered even if it's not in the communication range of the beacon. Otherwise, if the network is identified as mirrored, a transformation is performed, in which the ODs are changed with their $180\,^{\circ}$-complement.

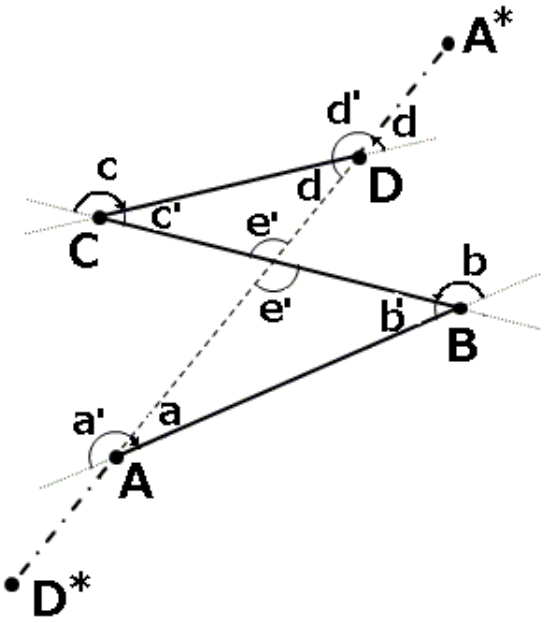
7.4.2 Special Cases

The orientation in some OD-paths can create false identification of mirrored path. Particularly those OD-path having OD-edges crossing over a straight line drawn between the source

and destination node. Figure 7.8 depicts a true OD-path and its mirrored OD-path, with one OD-edge crossing. When there is an odd number of crossings, the sum of changes in directions equals 0° for both true and mirrored OD-paths.



(a) True OD-path, 1 crossing



(b) Mirrored OD-path, 1 crossing

Figure 7.8: OD-path with One Edge Crossing

Proof. The sum of direction changes (angles a, b, c and d) from a true OD-path with one crossing, as shown in Fig. 7.8a, is equal to $0\,^\circ$:

$$a + b + c + d \;=\; 0\,^\circ$$

Let us consider the triangle interior angles sum theorem, which states that the sum of the interior angles of a triangle equals $180\,^\circ$:

$$\begin{aligned} a' + b' + e' &= 180\,^\circ \\ d' + c' + e' &= 180\,^\circ \end{aligned}$$

Therefore:

$$\begin{aligned} a' + b' + e' &= d' + c' + e' \\ a' + b' &= d' + c' \end{aligned} \tag{7.3}$$

Considering the supplementary angles in Fig. 7.8a, we have:

$$\begin{aligned} a' &= 180\,^\circ - a \\ b' &= 180\,^\circ - b \\ c' &= 180\,^\circ - c \\ d' &= 180\,^\circ - d \end{aligned}$$

Then Equation (7.3) can be rewritten as:

$$\begin{aligned} 180^\circ - a + 180^\circ - b &= 180^\circ - c + 180^\circ - d \\ a + b &= c + d \end{aligned} \tag{7.4}$$

Equation (7.4), shows that the magnitude of $(a+b)$ is equal to the magnitude of $(c+d)$. Additionally, from Fig. 7.8a it is possible to notice that angles c and d have opposite rotation direction than angles a and b. Therefore we can write that:

$$a + b = c + d = -(a + b)$$

Thus:

$$a + b + c + d = a + b - (a + b) = 0^\circ$$

□

Similarly, it can be proven that the sum of direction changes (angles a, b, c and d) in a mirrored OD-path with one OD-edge crossing, depicted in Fig. 7.8b, is also equal to 0°.

Proof. The sum of direction changes (angles a, b, c and d) from a mirrored OD-path with one crossing, is equal to 0°:

$$a + b + c + d = 0^\circ$$

Let us consider the triangle interior angles sum theorem:

$$\begin{aligned} d + c' + e' &= 180^\circ \\ a + b' + e' &= 180^\circ \end{aligned}$$

Therefore:

$$\begin{aligned} d + c' + e' &= a + b' + e' \\ d + c' &= a + b' \end{aligned} \tag{7.5}$$

Considering the supplementary angles in Fig. 7.8b, we have:

$$\begin{aligned} b' &= 180^\circ - b \\ c' &= 180^\circ - c \end{aligned}$$

Then Equation (7.5) can be rewritten as:

$$\begin{aligned} d + 180^\circ - c &= a + 180^\circ - b \\ d + b &= a + c \end{aligned} \tag{7.6}$$

Equation (7.6), shows that the magnitude of $(d + b)$ is equal to the magnitude of $(a + c)$. Additionally, from Fig. 7.8b it is possible to notice that angles a and c have opposite rotation direction than angles b and d. Therefore we can write that:

$$d + b = -(a + c)$$

Thus:

$$a + b + c + d = a + c - (a + c) = 0^\circ$$

□

The same can be proven for mirrored OD-paths with greater odd number of crossings (see Section 7.5.1).

The sum of direction changes is not affected when the number of crossing edges is even, for which it results as expected, 360° for true OD-paths, and 0° for mirrored OD-paths. Proofs for even crossings cases are provided in Section 7.5.2.

All these cases share a characteristic that is unaffected by the crossings parity (even, odd). Consider the number of changes in the angle sign for each change in direction in an OD-path. There are more sign changes in mirrored OD-paths than in true OD-paths. Changes in angles sign can be tracked in both of the candidate paths. The path having the least sign changes is selected as the true path. For example, let's consider both, true and mirrored, OD-paths presented in Fig. 7.8. Also, assume counter clockwise rotation as positive rotation. If we traverse the path from node A to node F in the true OD-path (Fig. 7.8a), then the sequence of direction changes is {a,b,-c,-d}. Thus, for this path there is only one sign change, whereas for the mirrored path (Fig. 7.8b the sequence of direction changes is {-a,b,-c,d}, resulting in three sign changes.

7.5 Additional Proofs for Special Cases

A discussion of special cases for the identification of mirrored topologies was provided in Section 7.4.2. In this Section, additional proofs supporting the successful identification of further special cases are presented.

7.5.1 Proof for Odd Crossings

Our objective is to prove that the sum of direction changes in an OD-path, having an odd number of OD-edge crossings (cx) over an imaginary straight line between source and destination, is equal to $0\,^{\circ}$.

7.5.1.1 OD-paths with $cx > 1$

Let us consider an OD-path with 3 crossings, Figure 7.9.

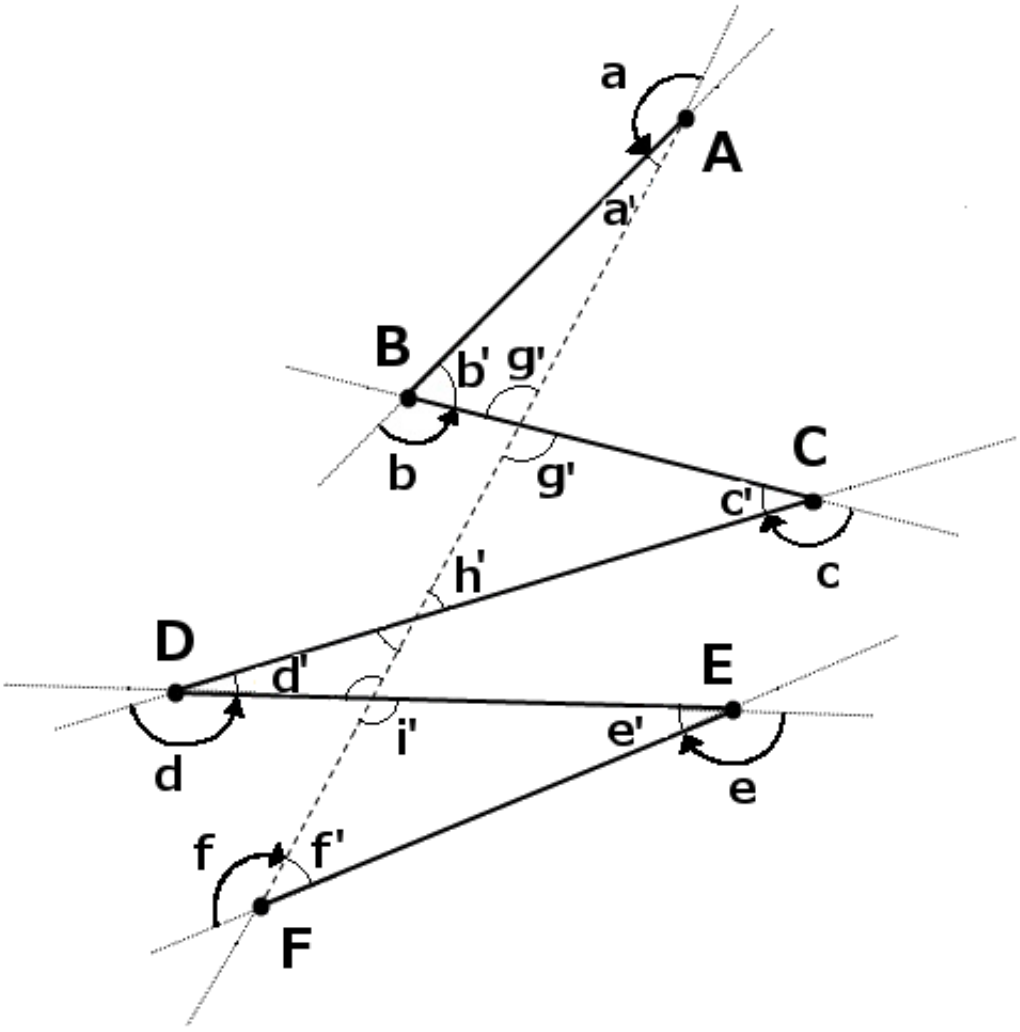


Figure 7.9: OD-path with 3 Edges Crossing

Proof. Sum of direction changes, angles a, b, c, d, e, f, from OD-path in Figure 7.9, is equal to $0\,^{\circ}$:

$$a + b + c + d + e + f = 0$$

Considering the triangle interior angles sum theorem:

$$\begin{aligned} a' + b' + g' &= 180\,^{\circ} \\ h' + c' + g' &= 180\,^{\circ} \\ h' + i' + d' &= 180\,^{\circ} \\ f' + i' + e' &= 180\,^{\circ} \end{aligned}$$

We can write:

$$\begin{aligned} a' + b' + g' &= h' + c' + f' \\ a' + b' &= h' + c' \end{aligned} \tag{7.7}$$

$$\begin{aligned} h' + i' + d' &= f' + i' + e' \\ h' + d' &= f' + e' \\ h' &= f' + e' - d' \end{aligned} \tag{7.8}$$

Replacing h' in Equation (7.7) with Equation (7.8) we obtain:

$$\begin{aligned} a' + b' &= f' + e' - d' + c \\ a' + b' + d &= f' + e' + c' \end{aligned} \tag{7.9}$$

Considering the supplementary angles:

$$\begin{aligned} a' &= 180^\circ - a \\ b' &= 180^\circ - b \\ c' &= 180^\circ - c \\ d' &= 180^\circ - d \\ e' &= 180^\circ - e \\ f' &= 180^\circ - f \end{aligned}$$

Then, Equation (7.9) can be rewritten as:

$$a + b + d = f + e + c \tag{7.10}$$

Equation (7.10), shows that the magnitude of $(a + b + d)$ is equal to the magnitude of $(f + e + c)$. Additionally, from Figure 7.9 it is possible to notice that angles c, e, and f have opposite rotation direction than angles a, b, and d. Considering the rotation direction we can write that:

$$f + e + c = -(a + b + d) \tag{7.11}$$

Then, the sum of the angles:

$$(a + b + d) + (f + e + c) = (a + b + d) - (a + b + d) = 0^\circ$$

□

7.5.2 Proof for Even Crossings

Our objective is to prove the following. First, the sum of direction changes in a true OD-path that has an even number of OD-edge crossings (cx) over an imaginary straight line between source and destination, is equal to 360 °. Second, the sum of direction changes is equal to zero for mirrored OD-paths with even number of OD-edge crossings.

7.5.2.1 OD-path with *cx = 2*

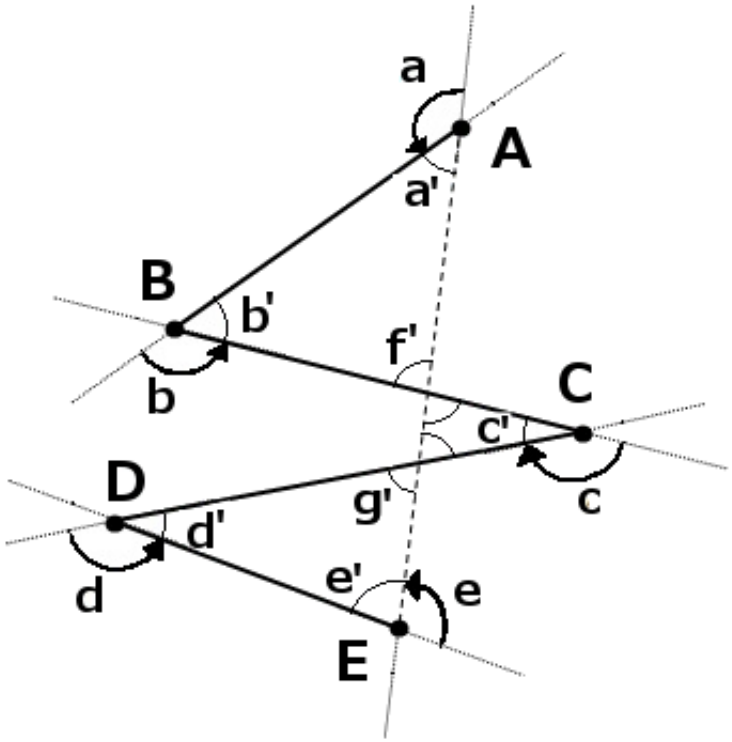


Figure 7.10: True OD-path with 2 Edges Crossing

Proof. The sum of direction changes (angles a, b, c, d and e) from a true OD-path with 2 OD-edges crossing, as shown in Fig. 7.10, is equal to 360 °:

$$a + b + c + d + e = 360\,^{\circ}$$

Let us consider the triangle interior angles sum theorem:

$$\begin{aligned} a' + b' + f' &= 180^\circ \\ f' + c' + g' &= 180^\circ \\ g' + e' + d' &= 180^\circ \\ g' &= 180 - d' - e' \end{aligned}$$

Therefore:

$$\begin{aligned} a' + b' + f' &= f' + c' + g' \\ a' + b' &= c' + g' \end{aligned} \tag{7.12}$$

Replacing g' in Equation (7.12):

$$\begin{aligned} a' + b' &= c' + 180 - d' - e' \\ a' + b' + d' + e' &= 180 + c' \end{aligned} \tag{7.13}$$

Considering the supplementary angles in Fig. 7.10, we have:

$$\begin{aligned} a' &= 180^\circ - a \\ b' &= 180^\circ - b \\ c' &= 180^\circ - c \\ d' &= 180^\circ - d \\ e' &= 180^\circ - e \end{aligned}$$

Then Equation (7.13) can be rewritten as:

$$\begin{aligned} a+b+d+e &= 360^\circ + c \\ c &= a+b+d+e-360^\circ \end{aligned}$$

From Fig. 7.10 it is possible to notice that angle c has opposite rotation direction than angles a, b, d, and e. Then, we can write:

$$c = -(a+b+d+e-360^\circ)$$

Therefore, the sum of the angles can be expressed as:

$$\begin{aligned} a+b+d+e+c &= a+b+d+e-a-b-d-e+360^\circ \\ a+b+d+e+c &= 360^\circ \end{aligned}$$

□

However, the sum of direction changes (angles a, b, c, d and e) in the mirrored OD-path with two OD-edges crossing, depicted in Fig. 7.11, is equal to 0°.

Proof. The sum of direction changes (angles a, b, c, d and e) from a mirrored OD-path with 2 OD-edges crossing is equal to 0°:

$$a+b+c+d+e = 0^\circ$$

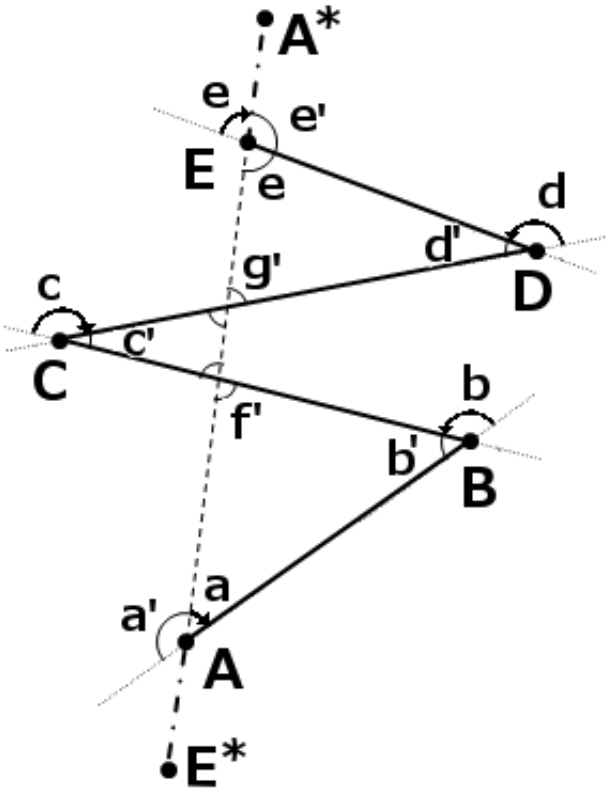


Figure 7.11: Mirrored OD-path with 2 Edges Crossing

Let us consider the triangle interior angles sum theorem:

$$\begin{aligned} e + d' + g' &= 180\,^\circ \\ c' + g' + f' &= 180\,^\circ \\ f' + b' + a &= 180\,^\circ \\ f' &= 180 - b' - a \end{aligned}$$

Therefore:

$$\begin{aligned} e + d' + g' &= c' + g' + f' \\ e + d' &= c' + f' \end{aligned} \tag{7.14}$$

Replacing f' in Equation (7.14):

$$\begin{aligned} e + d' &= c' + 180 - b' - a \\ a + e - c' &= 180 - d' - b' \end{aligned} \tag{7.15}$$

Considering the supplementary angles in Fig. 7.11, we have:

$$\begin{aligned} b' &= 180^\circ - b \\ c' &= 180^\circ - c \\ d' &= 180^\circ - d \end{aligned}$$

Then, Equation (7.15) can be rewritten as:

$$a + e + c = d + b$$

From Fig. 7.11 it is possible to notice that angles b and d have opposite rotation direction than angles a, e, and c. Therefore, we can write that:

$$d + b = -(a + e + c)$$

Hence,

$$\begin{aligned} a + b + d + e + c &= (a + e + c) - (a + e + c) \\ a + b + d + e + c &= 0^\circ \end{aligned}$$

□

7.5.2.2 OD-paths with $cx > 2$

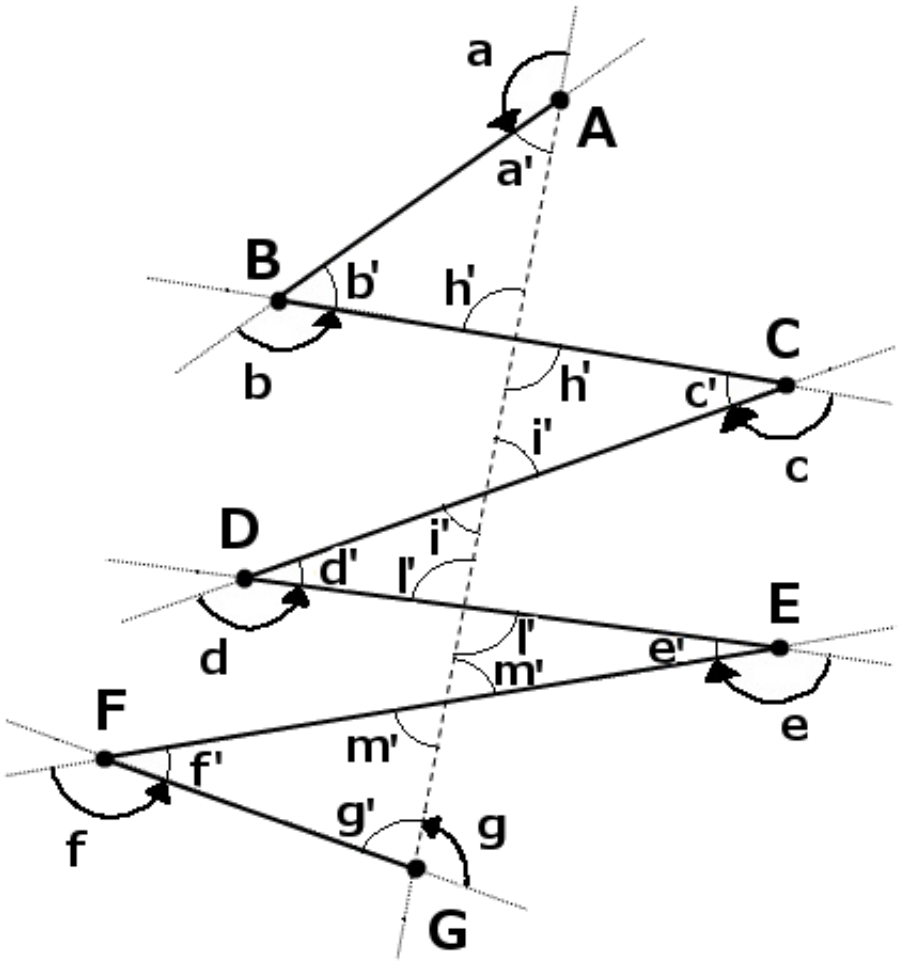


Figure 7.12: True OD-path with 4 Edges Crossing

OD-path with more even number of crossings can be proven to have the sum of direction changes equal to 360° in a similar manner to OD-path with 2 OD-edges crossing.

For instance, let us consider a true OD-path with four crossings, Fig. 7.12. It can be proved that:

$$a + b + d + f + g \;\; = \;\; 360^\circ + c + e$$

From the Fig. 7.12, c and e show to have opposite rotation direction than angles a, b, d, f, g. Considering the rotation direction we can write:

$$c + e \;=\; -(a + b + d + f + g - 360\,^\circ)$$

Consequently, the sum of the angles is equal to $360\,^\circ$:

$$\begin{aligned} & a + b + d + f + g + e + c \\ = \;& a + b + d + f + g - (a + b + d + f + g - 360\,^\circ) \\ = \;& 360\,^\circ \end{aligned}$$

We have proved that the sum of direction changes in OD-path with even number of crossings yields the same results as with OD-paths with no crossings: $360\,^\circ$ for true OD-paths, and $0\,^\circ$ for mirrored OD-paths.

7.6 Summary

A node's relative orientation presents a new alternative to solve wireless sensor network problems otherwise solved with localization technology that is sometimes difficult to incorporate due to network limitations. The solution presented in this chapter takes advantage of jumping nodes that spin while airborne, to obtain neighbor nodes' relative orientation. Results from field experiments show the viability of our approach. Our proposed algorithms can be adapted in a WSN to work in a distributed manner. Moreover, having an oriented network topology relieves the network from precise deployment of beacon nodes with location information, thereby, reducing the number of such nodes in the network.

Future areas of work will seek to enhance the proposed orientation algorithms to be more noise tolerant. Likewise, oriented networks present an opportunity to solve network-dynamics problems, such as obstacle and boundary identification.

Chapter 8

SUMMARY AND FUTURE WORK

8.1 Summary

This dissertation discussed several research topics addressing issues related to wireless sensor networks (WSN) connectivity and area coverage problems. First, in Chapter 3, a discussion of the primary factors that impact signal propagation of wireless sensors is provided and studied through experimental evaluations. Changes in sensor communication range are studied by varying sensors' heights relative to the surface. Results indicated that transmitting from higher than ground level increases the wireless communication range of the sensor.

The aforementioned findings served as motivation to propose a novel communication technique that relies on the jumping capabilities of sensors. While the jumping sensor robots are airborne, the change in elevation enhances their ability for a short time to successfully communicate with other sensors that are out of communication range at the ground level. Field experiments were conducted and results showed a considerable improvement in wireless communication ranges. An increase from 5m-10m to 30m-60m was seen when the sensor jumps to an average height of 1 meter. Additional, a two-way airborne communication process is defined and tested with sensor nodes, proving to be effective to communicate with distant receivers.

Second, in Chapter 4 the relation between network connectivity and area coverage is discussed. A new Hopping Sensor Network Model (HSNM) is introduced. The model high-

lights the factors that have to be taken into account in a jumping sensor network in order to increase the network connectivity, and thus increasing the network covered area. Furthermore, a Hopping Sensor Routing Protocol (HSRP) is designed from the HSNM to balance the energy consumption in the network. Simulation results showed a 20% network energy savings when HSRP was employed, resulting in a longer network utile life.

Third, in order to improve base station connectivity in a jumping sensor network, and to overcome communication holes, two decentralized algorithms for the discovery of isolated sensor nodes are defined in Chapter 5. The algorithms employ only those jumping sensor that are located on the boundary of the clusters to perform discovery, while saving network energy. The main difference between the discovery algorithms is on the selection of boundary nodes to perform discovery, one being more selective and energy preserving than the other. Simulation results demonstrated the efficiency of the algorithms to increase network connectivity, and to reduce the network energy consumption for discovery. From all the evaluated topology scenarios, the algorithms were able to achieve 100% network connectivity. With regards to network wide energy consumption for discovery tasks, results showed significant savings, 57% energy saving for *aggressive* discovery and 79% energy saving for *smart* discovery.

Fourth, Chapter 6 presents an unique fitness evaluation methodology for wireless sensor network topologies. Network connectivity is crucial for a network to be functional. Nonetheless, network responsiveness and its capacity to meet communication flow demands are important as well. This work evaluated network performance aspects related to network connectivity from the application's perspective. A Quality of Connectivity (QoC) performance metric is defined to evaluate the network connectivity in terms of how geographically distributed the sensor nodes are in the topology. The metric evaluates and assigns a score to the network topology, which serves as an indicator of how susceptible is the topology to

communication problems and latency.

Centralized and distributed algorithm approaches are presented to maximize network performance. Maximizing network performance with fewer operations becomes important in WSN where energy and computing resources are limiting factors. The algorithms make smart cluster merging operations attempting to yield maximum QoC for each operation. Network QoC improvement and the complexity of executing the approaches is evaluated in multiple and diverse network topology scenarios. From all the algorithm approaches, Distributed Best Merge Next from base station (DBMN-BS) was discussed to be less computing intensive than its centralized counterpart, Best Merge Next from base station (BMN-BS), having an equally outstanding network QoC yield per operation.

Last, a node's relative orientation presents a new alternative to solve wireless sensor network problems otherwise solved with localization technology that is sometimes difficult to incorporate due to network limitations. The solution presented in Chapter 7 takes advantage of jumping nodes that spin while airborne, to obtain neighbor nodes' relative orientation. Results from field experiments show the viability of our approach. Our proposed algorithms can be adapted in a WSN to work in a distributed manner. Moreover, having an oriented network topology relieves the network from precise deployment of beacon nodes with location information, thereby, reducing the number of such nodes in the network. Likewise, oriented networks present an opportunity to solve network-dynamics problems, such as obstacle and boundary identification.

8.2 Future Work

This research established the basis for sensor communications and exploration techniques that were beyond the available technology of half a decade ago. Jumping sensors provide an alternative to address common wireless sensor network problems. Moreover, the development of newer type of sensor robots will provide opportunities to deploy and study wireless sensor networks from different perspectives.

Future work is envisioned to further enhance and incorporate the proposed approaches aimed to improve network connectivity and area coverage with state of the art technology available today. In particular, the following topics are proposed to serve as future directions.

8.2.1 Heterogeneous Wireless Sensor Network Test-bed

Environmental monitoring with heterogeneous wireless sensor networks is an encouraging, yet challenging area of research. Heterogeneous networks are characterized for having diverse sets of specialized nodes, each presenting unique abilities, such as, the type of movement, making them suitable for adverse scenarios. Moreover, there has been an increase in autonomous vehicles research recently. It would be interesting to adapt such to a multi-robot test-bed, incorporating the use of jumping sensors. Furthermore, jumping sensor prototypes have advanced to the point where Airborne Sensor Communication (ASC) can start to be implemented and tested in further challenging scenarios.

The usage of multi-robot environments poses new research questions. For instance, from the deployment perspective, what factors should be considered to establish the quantity distribution for different types of sensor nodes? Additionally, how should special node units be allocated to needed areas?

As an initiative, the work introduced in Chapter 6 to increase networks performance by optimizing sensor nodes deployment can be extended to apply in heterogeneous sensor networks. Efficient node deployment should be re-defined to consider the diversity in nodes characteristic presented in such networks.

8.2.2 Neighbor Discovery

This work developed a jumping sensor neighbor discovery protocol, and enhanced it to be more energy efficient by reducing the selection of nodes to perform discovery (see Chapter 5). The enhanced algorithm performs discovery making use of two-hops boundary nodes neighbors of the initially selected nodes. It is motivating to focus research to optimize the selection of nodes to perform discovery, i.e., find the minimum discoverer set.

The discovery node reduction hypothesis relies on the fact that once a boundary node is selected to perform discovery, its communication airborne range grows (a factor of 6 on average for 1 meter jump height, see Chapter 3) such that the contribution of two-hop away boundary neighbors may still be minimal. Finding the minimum discoverer set is a difficult problem for wireless sensor nodes without GPS (Global Position Systems) capabilities, where there is not a precise sensor location. However, the availability of sensor's relative orientation (discussed in Chapter 7) allows for newer solutions to be explored.

BIBLIOGRAPHY

BIBLIOGRAPHY

802.15.4-IEEE-Standard (2006), 802.15.4-2006 IEEE Standard for Information technology - Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), . (Cited on page 87.)

Aeronautics and Space Engineering Board, N. R. C. (1995), *The Global Positioning System: A Shared National Asset: Recommendations for Technical Improvements and Enhancements*, National Academy Press, URL `http://books.google.com/books?id=FAHk65slfY4C`. (Cited on page 23.)

Aguayo, D., Bicket, J., Biswas, S., Judd, G. and Morris, R. (2004), Link-level measurements from an 802.11b mesh network, *SIGCOMM Comput. Commun. Rev.*, vol. 34 (4), p. 121–132. (Cited on page 12.)

Aissani, M., Mellouk, A., Badache, N. and Djebbar, M. (2008), A New Approach of Announcement and Avoiding Routing Voids in Wireless Sensor Networks, in Global Telecommunications Conference. GLOBECOM 2008. IEEE, p. 1–5. (Cited on page 66.)

Balakrishnan, H., Baliga, R., Curtis, D., Goraczko, M., Miu, A., Priyantha, B., Smith, A., Steele, K., Teller, S. and Wang, K. (2003), Lessons from Developing and Deploying the Cricket Indoor Location System, Preprint., Massachusetts Institute of Technology. (Cited on page 24.)

Biaz, S., Yiming, J., Qi, B. and Wu, S. (2005), Dynamic signal strength estimates for indoor wireless communications, in Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '05), vol. 1, p. 602–605. (Cited on page 12.)

Burdick, J. and Fiorini, P. (2003), Minimalist Jumping Robots for Celestial Exploration, *The International Journal of Robotics Research*, vol. Vol. 22 (No. 7-8), p. 653–674. (Cited on page 21.)

Camp, T., Boleng, J. and Davies, V. (2002), A Survey of Mobility Models for Ad Hoc Network Research, *Wireless Communications & Mobile Computing (WCMC): Special Issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, p. 483–502. (Cited on page 14.)

Carbunar, B., Grama, A., Vitek, J. and Carbunar, O. (2006), Redundancy and Coverage Detection in Sensor Networks, *ACM Transactions on Sensor Networks*, vol. 2 (1), p. 35. (Cited on page 11.)

Cen, Z. and Mutka, M. W. (2008), Relocation of Hopping Sensors, in ICRA '08: Proceedings of The IEEE International Conference on Robotics and Automation, p. 569–574. (Cited on pages 22 and 115.)

Cheng, Y.-M. and Yen, L.-H. (2006), Range-Based Density Control for Wireless Sensor Networks, in Proceedings of the 4th Annual Communication Networks and Services Research Conference (CNSR '06), p. 170–180, Washington, DC, USA. (Cited on page 11.)

Cintrón, F. J. and Mutka, M. W. (2010), Hopping Enhanced Sensors for Efficient Sensor Network Connectivity and Coverage, in MASS '10: Proceedings of the 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems, San Francisco. (Cited on pages 89 and 120.)

Cintrón, F. J., Pongaliur, K., Mutka, M. W. and Xiao, L. (2009), Energy Balancing Hopping Sensor Network Model to Maximize Coverage, in ICCCN '09: Proceedings of The IEEE 18th International Conference on Computer Communications and Networks, p. 1–6. (Cited on page 85.)

Cintrón, F. J., Pongaliur, K., Mutka, M. W., Xiao, L., Zhao, J. and Xi, N. (2012), Leveraging Height in a Jumping Sensor Network to Extend Network Coverage, *Wireless Communications, IEEE Transactions on*, vol. 11 (5), p. 1840 –1849. (Cited on page 85.)

Ding, Y., Wang, C. and Xiao, L. (2007), A connectivity based partition approach for node scheduling in sensor networks, in DCOSS'07: Proceedings of the 3rd IEEE international conference on Distributed computing in sensor systems, p. 354–367, Springer-Verlag, Berlin, Heidelberg. (Cited on page 12.)

Du, X., Mandala, D., Zhang, W., You, C. and Xiao, Y. (2007), A Boundary-Node based Localization Scheme for Heterogeneous Wireless Sensor Networks, in Military Communications Conference. MILCOM 2007. IEEE, p. 1–7. (Cited on page 66.)

Dyo, V., Ellwood, S. A., Macdonald, D. W., Markham, A., Mascolo, C., Pásztor, B., Scellato, S., Trigoni, N., Wohlers, R. and Yousef, K. (2010), Evolution and Sustainability of a Wildlife Monitoring Sensor Network, in SenSys '10: Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, p. 127–140, ACM, Zurich, Switzerland, URL `http://doi.acm.org/10.1145/1869983.1869997`. (Cited on page 90.)

Fayed, M. and Mouftah, H. T. (2009), Localised convex hulls to identify boundary nodes in sensor networks, *Int. J. Sen. Netw.*, vol. 5 (2), p. 112–125. (Cited on page 65.)

German, J. (2000), Hop to it: Sandia hoppers leapfrog conventional wisdom about robot mobility, *Sandia Lab News*, vol. 52 (21), URL `http://www.sandia.gov/LabNews/LN10-20-00/hop_story.html`. (Cited on page 21.)

Goldsmith, A. (2005), *Wireless Communications*, Cambridge University Press. (Cited on page 24.)

Green, D. and Obaidat, A. (2002), An accurate line of sight propagation performance model for ad-hoc 802.11 wireless LAN (WLAN) devices, in ICC '02: Proceedings of The IEEE International Conference on Communications, vol. 5, p. 3424 – 3428 vol.5. (Cited on pages 30 and 31.)

Han, Z. and Poor, H. (2009), Coalition Games with Cooperative Transmission: A Cure for the Curse of Boundary Nodes in Selfish Packet-Forwarding Wireless Networks, *Communications, IEEE Transactions on*, vol. 57 (1), p. 203–213. (Cited on page 66.)

Hata, M. (1980), Empirical formula for propagation loss in land mobile radio services, *IEEE Transactions on Vehicular Technology*, vol. 29 (3), p. 317–325, URL `http://dx.doi.org/10.1109/T-VT.1980.23859`. (Cited on page 32.)

Howard, A., Mataric, M. and Sukhatme, G. S. (2002), Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem, in DARS '02: Proceedings of The Sixth International Symposium on Distributed Autonomous Robotics Systems, p. 299–308, Fukupka, Japan, URL `http://cres.usc.edu/pubdb_html/files_upload/71.pdf`. (Cited on page 13.)

Kavak, A., Yang, W., Dandekar, K. R. and Xu, G. (1999), Effects of base station antenna height and mobile terminal movement on the vector propagation channels, *IEEE 49th Vehicular Technology Conference (VTC 1999)*, vol. 1, p. 777–781 vol.1. (Cited on page 12.)

Kim, M. and Mutka, M. W. (2009a), Multipath-based relocation schemes considering balanced assignment for hopping sensors, in IROS'09: Proceedings of the IEEE/RSJ international conference on Intelligent robots and systems, p. 5095–5100, IEEE Press, Piscataway, NJ, USA. (Cited on page 22.)

Kim, M. and Mutka, M. W. (2009b), On Relocation of Hopping Sensors for Balanced Migration Distribution of Sensors, in Proceedings of the International Conference on Computational Science and Its Applications: Part II, ICCSA '09, p. 361–371, Springer-Verlag, Berlin, Heidelberg, URL `http://dx.doi.org/10.1007/978-3-642-02457-3_31`. (Cited on page 22.)

Kim, M., Mutka, M. W. and Choo, H. (2010), On Relocation of Hopping Sensors for Rugged Terrains, in ICCSA '10: Proceedings of the International Conference on Computational Science and its Applications, vol. 0, p. 203–210, IEEE Computer Society, Los Alamitos, CA, USA. (Cited on page 22.)

Kovac, M., Fuchs, M., Guignard, A., Zufferey, J.-C. and Floreano, D. (2008), A miniature 7g jumping robot, in ICRA '08: Proceedings of the IEEE International Conference on Robotics and Automation, p. 373–378. (Cited on page 21.)

Lee, W. C. Y. (1986), *Mobile Communications Design Fundamentals*, Howard W. Sams & Co., Indianapolis, IN, USA. (Cited on page 30.)

Li, N., Hou, J. C. and Sha, L. (2003), Design and Analysis of an MST-based Topology Control Algorithm, in INFOCOM '03. Proceedings of The Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, p. 1702–1712. (Cited on page 18.)

Lin, C.-Y., Peng, W.-C. and Tseng, Y.-C. (2006), Efficient in-network moving object tracking in wireless sensor networks, *Mobile Computing, IEEE Transactions on*, vol. 5 (8), p. 1044–1056. (Cited on page 90.)

Liu, C.-M. and Lee, C.-H. (2004), Power efficient communication protocols for data gathering on mobile sensor networks, *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, vol. 7, p. 4635–4639. (Cited on page 11.)

Liu, C.-M., Lee, C.-H. and Wang, L.-C. (2004), Power-efficient communication algorithms for wireless mobile sensor networks, in PE-WASUN '04: Proceedings of the 1st ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks, p. 121–122, ACM, NY, USA. (Cited on page 11.)

Longley, A. and Rice, P. (1968), Prediction of Tropospheric Radio Transmission over Irregular Terrrain, A Computer Method, Technical report, ESSA Tech. Rep. ERL 79-ITS 67, Washington, DC. (Cited on page 32.)

Mogensen, P. E., Jensen, C. and Andersen, J. B. (1991), 1800 MHz Mobile Net Planning Based based on 900 MHz Measurements, in COST 231 TD(91)-008, University of Aalborg, Firenze. (Cited on page 32.)

Pei, Y., Cintrón, F. J., Mutka, M. W., Zhao, J. and Xi, N. (2009), Hopping Sensor Relocation in Rugged Terrain, in The IEEE/RSJ International Conference on Intelligent RObots and Systems (IROS 2009), . (Cited on pages 22 and 115.)

Priyantha, N. B., Miu, A. K., Balakrishnan, H. and Teller, S. (2001), The Cricket Compass for Context-Aware Mobile Applications, in 7th ACM MOBICOM, Rome, Italy. (Cited on page 24.)

Ramanathan, R. and Rosales-Hain, R. (2000), Topology control of multihop wireless networks using transmit power adjustment, in INFOCOM '00: Proceedings of The Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies., vol. 2, p. 404–413. (Cited on pages 13, 16 and 80.)

Römer, K. and Mattern, F. (2004), The Design Space of Wireless Sensor Networks, *IEEE Wireless Communications*, vol. 11 (6), p. 54–61. (Cited on page 9.)

Sahoo, P., Hsieh, K.-Y. and Sheu, J.-P. (2007), Boundary Node Selection and Target Detection in Wireless Sensor Network, in Wireless and Optical Communications Networks. WOCN '07. IFIP International Conference on, p. 1–5. (Cited on page 65.)

Santi, P. (2005), *Topology control in wireless ad hoc and sensor networks*, John Wiley & Sons Ltd. (Cited on page 15.)

Scarfogliero, U., Stefanini, C. and Dario, P. (2007), Design and Development of the Long-Jumping "Grillo" Mini Robot, in ICRA '07: Proceedings of the IEEE International Conference on Robotics and Automation, p. 467–472. (Cited on page 21.)

Shah, R. and Rabaey, J. (2002), Energy aware routing for low energy ad hoc sensor networks, in Wireless Communications and Networking Conference. WCNC2002. IEEE, vol. 1, p. 350 – 355. (Cited on page 11.)

Song, W.-Z., Huang, R., Xu, M., Ma, A., Shirazi, B. and LaHusen, R. (2009), Air-dropped sensor network for real-time high-fidelity volcano monitoring, in MobiSys '09: Proceedings of the 7th international conference on Mobile systems, applications, and services, p. 305–318, ACM, New York, NY, USA. (Cited on pages 24, 64 and 83.)

TIA TR8 Working Group 8.8 Technology Compatibility (1997), A Report on Technology Independent Methodology for the Modeling, Simulation and Empirical Verification of Wireless Communications System Performance in Noise and Interference Limited Systems Operating on Frequencies between 30 and 1500MHz, Technical report, TR-8 Mobile and Personal Private Radio Standards Committee, TELECOMMUNICATIONS INDUSTRY ASSOCIATION (TIA) Mobile and Personal Communications, IEEE Vehicular Technology Society Propagation Committee. (Cited on page 32.)

Wang, X., Xing, G., Zhang, Y., Lu, C., Pless, R. and Gill, C. (2003), Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks, in SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems, p. 28–39, ACM, New York, NY, USA. (Cited on page 12.)

Warneke, B., Last, M., Liebowitz, B. and Pister, K. (2001), Smart Dust: communicating with a cubic-millimeter computer, *Computer*, vol. 34 (1), p. 44–51. (Cited on page 22.)

Wei, T. E., Nelson, G. M., Quinn, R. D., Verma, H. and Garverick, S. L. (2000), Design of a 5-cm monopod hopping robot, in ICRA '00: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 3, p. 2828–2833 vol.3. (Cited on page 21.)

Whitehouse, C. D. (2002), The design of calamari: an ad-hoc localization system for sensor networks, Master's thesis, University of California at Berkeley. (Cited on page 24.)

Wu, X. and Liu, M. (2012), In-situ soil moisture sensing: measurement scheduling and estimation using compressive sensing, in IPSN '12: Proceedings of the 11th international conference on Information Processing in Sensor Networks, p. 1–12, ACM, Beijing, China, URL `http://doi.acm.org/10.1145/2185677.2185679`. (Cited on page 90.)

Yanmaz, E. (28-30 April 2008), Impact of Topology-dependent and independent Mobility Models on the Connectivity of Wireless Networks, *IEEE Sarnoff Symposium*, p. 1–5. (Cited on page 16.)

Younis, M., Munshi, P., Gupta, G. and Elsharkawy, S. M. (2006), On Efficient Clustering of Wireless Sensor Networks, in DSSNS '06: Proceedings of the Second IEEE Workshop on Dependability and Security in Sensor Networks and Systems, p. 78–91. (Cited on page 12.)

Zhang, H. and Hou, J. C. (2005), Maintaining Sensing Coverage and Connectivity in Large Sensor Networks, *Ad Hoc & Sensor Wireless Networks*, vol. 1 (1-2), p. 89–124, URL http://www.oldcitypublishing.com/AHSWN/AHSWN1.1-2abstracts/Zhangabs.html. (Cited on page 12.)

Zhao, J., Yang, R., Xi, N., Gao, B., Fan, X., Mutka, M. W. and Xiao, L. (2009), Development of a Miniature Self-stabilization Jumping Robot, in IROS '09: Proceedings of The IEEE/RSJ International Conference on Intelligent RObots and Systems, p. 2217–2222. (Cited on pages xiii, 4, 5, 22 and 71.)

Zhao, J., Xi, N., Gao, B., Mutka, M. W. and Xiao, L. (2010), Design and Testing of a Controllable Miniature Jumping Robot, in IROS '10: Proceedings of The IEEE/RSJ International Conference on Intelligent Robots and Systems, . (Cited on page 22.)

Zhao, J., Xu, J., Gao, B., Xi, N., Cintron, F. J., Mutka, M. W. and Xiao, L. (2013), MSU Jumper: A Single-Motor-Actuated Miniature Steerable Jumping Robot, *IEEE transactions on Robotics*, vol. 29 (3), p. In Press. (Cited on page 85.)

Zou, Y. and Chakrabarty, K. (2003), Sensor Deployment and Target Localization Based on Virtual Forces, in INFOCOM '03: Proceedings of The Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 2, p. 1293–1303 vol.2. (Cited on page 13.)