

ABSTRACT

Title of dissertation: **SALAM: A SCALABLE ANCHOR-FREE
LOCALIZATION ALGORITHM FOR
WIRELESS SENSOR NETWORKS**

Adel Amin Abdel Azim Youssef, Doctor of Philosophy, 2006

Dissertation directed by: **Professor Ashok K. Agrawala
Department of Computer Science**

In this dissertation, we present SALAM, a scalable anchor-free protocol for localization in wireless sensor networks. SALAM can determine the positions of sensor nodes without any infrastructure support. We assume that each node has the capability to estimate distances to its corresponding neighbors, that are within its transmission range. SALAM allows to trade the accuracy of the estimated position against node transmission range and/or computational power. The application layer can choose from a whole range of different options, to estimate the sensor node's positions with different accuracy while conserving battery power.

Scalability is achieved by dividing the network into *overlapping* multi-hop clusters each with its own cluster head node. Each cluster head is responsible for building a local relative map corresponding to its cluster using intra-cluster node's range measurements. To obtain the global relative topology of the network, the cluster head nodes collaboratively combine their local maps using simple matrix transformations.

In order for two cluster heads to perform a matrix transformation, there must be at

least three *boundary nodes* that belongs to both clusters (i.e. the two clusters are overlapping with degree ≥ 3). We formulate the overlapping multi-hop clustering problem and present a randomized distributed heuristic algorithm for solving the problem. We evaluate the performance of the proposed algorithm through analytical analysis and simulation.

A major problem with multi-hop relative location estimation is the error accumulated in the node position as it becomes multi-hop away from the cluster head node. We analyze different sources of error and develop techniques to avoid these errors. We also show how the local coordinate system (LCS) affects the accuracy and propose different heuristics to select the LCS.

Simulation results show that SALAM achieves precise localization of sensors. We show that our approach is scalable in terms of communication overhead regardless of the network size. In addition, we capture the impact of different parameters on the accuracy of the estimated node's positions. The results also show that SALAM is able to achieve accuracy better than the current ad-hoc localization algorithms.

SALAM: A SCALABLE ANCHOR-FREE LOCALIZATION
ALGORITHM FOR WIRELESS SENSOR NETWORKS

by

Adel Amin Abdel Azim Youssef

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2006

Advisory Committee:

Professor Ashok K. Agrawala, Chair/Advisor
Professor Eyad Abed, Dean's Representative
Professor Raymond Miller
Professor A. Udaya Shankar
Associate Professor Amitabh Varshney

© Copyright by
Adel Amin Abdel Azim Youssef
2006

To My Parents, My Wife, Habiba and Maryam

ACKNOWLEDGMENTS

Thanks to Allah the exalted, the most merciful, for giving me the strength and persistence to keep going with this research even during the most difficult moments. May Allah accept this work and count it as a good deed and help me to benefit the whole world with the knowledge I obtain.

My deep gratefulness to my country Egypt where I grew up and learned the first lessons and experiences in my life, and I hope that I can pay it back some day. My thanks are as well for University of Maryland where I learned how to do research. I gratefully acknowledge the support of the computer science department family because of whom my graduate experience has been one that I will cherish forever.

I am indebted to my advisor Professor Ashok Agrawala for giving me an invaluable opportunity to work on challenging and extremely interesting projects over the past years. He has always made himself available for help and advice and there has never been an occasion when I've knocked on his door and he hasn't given me time. It has been a pleasure to work with and learn from such an extraordinary individual.

I would also like to express my deepest thanks to Dr. Mohamed Younis for his continuous support during my Ph.D. No doubt, you are the maestro of my research work. Your ever valuable feedback, enthusiasm and friendship have helped me a lot. I would like also to thank my dearest brother, Dr. Moustafa Youssef, for his invaluable guidance and support through the long trip. He was always there for advice, and help. Jazakom

Allah Khayran both of you.

Special thanks are due to Dr. Udaya Shankar, Dr. David Mount, Dr. Samir Khuller, Dr. Raymond Miller, and Dr. Amitabh Varshney, for their advice and support. They were always educating and available when I need them.

I owe my deepest thanks to my parents - my mother *Sorya* and my father *Amin* who have always stood by me and guided me through my career, and have pulled me through against impossible odds at times. Words cannot express the gratitude I owe them. You did and still do lots to me. I would like also to extend my gratefulness to my parents in law for their prayers for me and support.

These acknowledgements would not be complete without expressing my gratitude and love to my wife Aliaa. Whatever I say will never express my feelings, appreciation, respect towards her and ever love. I think, for a married (with children) Ph.D. student, the degree is unattainable unless a very cooperative and helpful wife is there. Aliaa, you deserve this degree with me, actually you deserve even a higher degree. May Allah reward you for your efforts and patience.

This work would never come to an end without the two little angels by my side, my daughters, Habiba and Maryam. With their smiles I managed to survive very dark moment in my life. I love you so much and I hope you forgive me for being away most of the time in the last couple of months. I promise I will try from now on to be always with you and compensate you for this tough period we all had.

I would also like to thank my dearest friends, and brothers in Islam, Mohamed Abdallah, and Tamer El Sharnouby. I love you for the sake of Allah. They were always supporting me with continuous guidance, feedback, and encouragement.

I am very grateful to the Tauba Halaqa community, where I gained lots of knowledge and experiences, much more beyond my Ph.D. in different subjects of life. I think the word "lots" does not express the quantity and the quality of the knowledge I learned from you. We, Youssef's family, love you and will never forget you. You will always be in our hearts forever. We love you for the sake of Allah. I hope that Allah SWT gather us all in a superior place.

Finally, I would like to thank Mr. Amr Khaled and Dr. Tareq Suidan for helping me understand the objective of this life. With their invaluable lectures, I managed to organize my priorities not only in the PhD but also through the long trip to a better life.

It is impossible to remember all, and I apologize to those I have inadvertently left out.

TABLE OF CONTENTS

1	Introduction	1
1.1	Wireless Sensor Networks	1
1.2	The Location Discovery Problem	3
1.3	Proposed Solution	7
1.4	Contributions	9
1.5	Dissertation Organization	10
2	A Taxonomy of Localization Schemes for Wireless Sensor Networks	12
2.1	Taxonomy Features	12
2.1.1	Anchor-based versus Anchor-free	13
2.1.2	Range Estimation Method	14
2.1.3	Range Combining Technique	15
2.1.4	Computational Model	17
2.1.5	Accuracy	17
2.1.6	Communication Power	18
2.1.7	Computation Power	18
2.1.8	Scalability	19
2.1.9	Capital Cost	19
2.1.10	Limitations	19
2.2	An Overview of Location Discovery Algorithms	20
2.2.1	Hop-TERRAIN	21
2.2.2	APS Algorithm	22
2.2.3	GPS-less	23
2.2.4	Convex Position Estimation	23
2.2.5	Iterative Multilateration	24
2.2.6	Collaborative Multilateration	25
2.2.7	GPS-free	26
2.2.8	MDS-MAP	27
2.2.9	Improved MDS-MAP	28
3	SALAM Overview	29
3.1	Introduction	29
3.2	System Model	31
3.3	Phase I: Range Estimation and Cluster Formation	32
3.4	Phase II: Local Location Discovery (LLD)	34
3.5	Phase III: Global Location Discovery (GLD)	36

4	The Overlapped K-hop (OK) Clustering Algorithm	39
4.1	Introduction	39
4.2	Related Work	43
4.3	Problem Formulation	48
4.3.1	Definitions	49
4.3.2	The Overlapped K-hop (OK) Clustering Problem	52
4.4	The OK Protocol Architecture	55
4.4.1	Data Structures	58
4.4.2	Messages	59
4.4.3	Timers	60
4.5	Performance Evaluation	61
4.5.1	Coverage, Cluster Overlapping and Connectivity Ratio	70
4.5.2	Cluster Size	77
4.5.3	Scalability	79
4.6	Analysis of The Results	84
4.6.1	Assumptions	86
4.6.2	Average Cluster Size	90
4.6.3	Average Overlapping Degree	91
4.6.4	Overall Communication Overhead	94
4.7	Correctness and Complexity	97
5	The Local Location Discovery Phase	102
5.1	Problem Definition	103
5.2	The Local Coordinate System (LCS)	105
5.3	Relative Position Estimation Using Three Distances	106
5.4	Multi-hop Relative Position Estimation	108
5.5	Relative Position Estimation Using Two Distances	110
5.6	The Reflection Error	111
5.7	Heuristics to Limit Error Accumulation	112
5.7.1	Selecting The Local Coordinate System (LCS)	113
5.7.2	Resolving Reflection	117
5.8	The Multi-hop Relative Location Estimation (MRLE) Algorithm	118
5.8.1	Definitions and Terminologies	118
5.8.2	The MRLE Algorithm	120
5.9	The Refinement Step	120
5.10	Validation and Performance Evaluation	121
5.10.1	Experiments Setup and Goals	121
5.10.2	The Effect of Local Coordinate System (LCS) on Performance	123
5.10.3	Achievable Accuracy	126
5.10.4	Optimization Factors	131
6	The Global Location Discovery Phase	134
6.1	The Best Transformation Matrix Problem	135
6.2	The Overlapping Graph	136
6.3	The Best Order of Transformations Problem	138

6.4	The Spanning Tree of The Overlapping graph	140
6.5	The GLD Algorithm	142
6.6	Validation and Performance Evaluation	145
6.6.1	Experiments Setup and Goals	145
6.6.2	The Effect of Spanning Tree on Accuracy	148
6.6.3	The Effect of Spanning Tree on Communication Overhead	149
6.6.4	Achievable Accuracy	151
6.7	Comparison With Other Localization Techniques	152
7	Conclusions and Research Directions	158
7.1	Research Directions	162
A	Area of Intersection Between Two Identical Circles	165
	Bibliography	166

LIST OF TABLES

4.1	Events summary of the OK clustering algorithm	62
5.1	Time complexity of different heuristics to select the local coordinate system (LCS)	117
5.2	Trading accuracy with computational power and transmission power . . .	132

LIST OF FIGURES

2.1	Range Combining Techniques: (a) Trilateration, (b) Triangulation, (c) Multilateration	15
3.1	Organizing the sensor network into overlapping multi-hop clusters	33
3.2	Building local coordinate system (LCS) within each cluster	35
3.3	Global network map can be obtained from local maps using simple matrix transformations	36
4.1	Illustrative Example	51
4.2	Flowchart of the OK cluster formation algorithm	57
4.3	Finite state machine of the OK protocol	63
4.4	The OK Algorithm	64
4.5	The relation between cluster head prob. (p) and percentage of covered nodes	71
4.6	The impact of average node degree (d) and cluster radius (k) on percentage of covered nodes	72
4.7	The cluster head prob. (p) has no effect on the average overlapping degree (AOD)	73
4.8	The impact of average node degree (d) and cluster radius (k) on average overlapping degree	74
4.9	The effect of the cluster head prob. (p) and average node degree (d) on percentage of connected clusters	76
4.10	The cluster head prob. (p) has no effect on cluster size properties	78
4.11	The effect of average node degree on cluster size properties	80
4.12	The effect of cluster radius on cluster size properties	81
4.13	The effect of different simulation parameters on communication overhead per node	83
4.14	Increasing the network size n does not effect the communication overhead	85
4.15	Circle representation of clusters	87
4.16	The relation between the analytical model for average cluster size (N_c) and simulation results	91
4.17	The relation between the analytical model for average number of edges per cluster (E_c) and simulation results	92
4.18	Overlapping Degree (O) between two overlapping clusters	92
4.19	The relation between the analytical model for average overlapping degree (AOD) and simulation results	94
4.20	The CH_AD message will follow a spanning tree rooted at the CH node ($k = 5$)	96

4.21	The relation between the analytical model for overall communication overhead per node and simulation results	98
5.1	The local coordinate system (LCS)	105
5.2	Estimating the position of a node (u) using three distances	107
5.3	Propagation of node position estimating starting from the reference nodes and moving towards the border of the cluster.	109
5.4	Estimating the position of a node (u) using two distances	110
5.5	The reflection propagation phenomena.	112
5.6	Different cases for skinny triangles	113
5.7	The cluster-head (CH) node is on the border of the cluster ($k=2$)	114
5.8	The Multi-hop Relative Location Estimation (MRLE) Algorithm	119
5.9	The effect of local coordinate system (LCS) on accuracy for different values of k and d	124
5.10	The effect of local coordinate system (LCS) on convergence latency	125
5.11	The effect of different simulation parameters on accuracy	127
5.12	The effect of different simulation parameters on convergence latency	128
5.13	The effect of CLIQUE factor (CF) on accuracy	129
5.14	The relation between CLIQUE factor (CF) and node transmission range (T_r)	130
5.15	The effect of CLIQUE Factor (CF) on convergence latency	130
5.16	The accuracy before and after optimization using MIE to select LCS	133
5.17	A comparison between accuracy before and after optimization using MIE and MET for selecting the LCS	133
6.1	The overlapping graph	137
6.2	Different methods for finding the spanning tree of the overlapping graph	141
6.3	An example of GLD algorithm	144
6.4	The effect of spanning tree weight on accuracy	149
6.5	The effect of spanning tree height on accuracy	150
6.6	The effect of spanning tree weight on communication overhead per node	151
6.7	The effect of spanning tree height on communication overhead per node	152
6.8	The effect of GLD phase on accuracy	153
6.9	The effect of average overlapping degree (AOD) on accuracy	153
6.10	A comparison between SALAM and MDS-based algorithms using uniform topology	155
6.11	A comparison between SALAM and MDS-based algorithms using GRID topology	156
A.1	Area of intersection of two circles	165

Chapter 1

Introduction

1.1 Wireless Sensor Networks

Recent advances in micro-electro-mechanical systems (MEMS) technology, digital electronics, and wireless communications have led to the development of inexpensive, low-power micro sensor nodes. These tiny sensor nodes that are capable of sensing, data processing, and communicating with each other, leverage the idea of sensor networks. A sensor network is a network composed of a large number of sensor nodes that are densely deployed either inside the phenomenon or very close to it. Sensor networks promise a significant improvement over the traditional sensing methods. The large-scale dense deployment extends the spatial coverage and achieves higher resolution, and increases the fault-tolerance and robustness of the system.

Sensor networks have wide applications including natural habitat monitoring [22, 39, 40], environmental observation, collecting information in disaster prone areas, military, medical, and surveillance applications. Networked sensors can warn about smoke on a remote forest indicating that a fire is about to start, or alternatively alert the possibility

of potential flooding by measuring rainfall and water level information [1]. In military, sensor networks can be used in battlefield surveillance, monitoring friendly forces, and target-tracking systems. In health, sensor nodes can also be deployed to telemonitor patients physiological data, track and monitor of doctors and patients inside a hospital. Embedding wireless biomedical sensors inside human body can be used to monitor Glucose level, detect Cancer and monitor general health. Biomedical sensor network is currently a very active research area although many challenges still exist [78]. Some other commercial applications include managing inventory, signal a machine malfunction to the control center in a factory, monitoring product quality, home automation, and smart home/office environments [83].

In general sensor networks classify as ad-hoc networks, especially when deterministic placement of nodes is not possible. However, ad-hoc networks have mostly been studied in the context of high mobility, high power nodes, and moderate network sizes. Although, many protocols and algorithms have been proposed for traditional wireless ad-hoc networks, sensor networks have unique features and application requirements, which make those protocols not well suited. To illustrate this point, the differences between sensor networks and ad-hoc networks are summarized as follows [2]:

- **Network size.** The number of sensor nodes in a sensor network can be several orders of magnitude higher than the number of nodes in an ad-hoc network.
- **Node density.** Sensor nodes are densely deployed. In general, the density can be as high as 20 sensor nodes/m³ [81].
- **Node capabilities.** Sensor nodes are limited in power, computational capacities,

and memory.

- **Communication model.** Sensor nodes mainly use a broadcast communication paradigm, whereas most ad-hoc networks are based on point-to-point communications. Moreover, since large number of nodes is densely deployed, neighbor nodes may be very close to each other. Hence, *multihop* communication in sensor network is expected to consume less power than single hop communication. Furthermore, the transmission power levels can be kept low.
- **Topology.** The topology of a sensor network changes very frequently mainly due to node failure. Mobility is another factor that may lead to topology changes although in sensor networks, many applications assume that the network is *stationary*.
- **Self-organization.** The position of sensor nodes needs not to be engineered or predetermined. Hence, sensor network protocols and algorithms must possess self-organizing capabilities.

These unique features of sensor networks have raised some interesting challenges that must be considered when designing a protocol or an algorithm for sensor networks. These design challenges are addressed in the next section.

1.2 The Location Discovery Problem

Sensor nodes can be either thrown in as a mass, deployed by dropping from a plane, delivered in an artillery shell, missile, or simply placed one by one by either a human or a robot. Knowledge of node location in such randomly deployed networks is an essential

requirement for many applications. Naturally, a sensor node needs to possess knowledge of its physical location to report the geographical origin of events. For example in target tracking applications the sensor readings have to be correlated to the sensor position in order to locate the target. Moreover, in large-scale sensor networks, localization is commonly used for routing scalability. Geographic-aware routing algorithms such as GEDIR [84], or geocast [67], maintain reduced or no routing tables at all which matches well the limited memory of the sensor nodes. The location of the nodes can also be used to study the coverage properties of the network [41]. Furthermore, it can be used to query nodes over a specific geographical area. Sensor position can also serve as a unique node identifier, making it unnecessary for each sensor to have a unique ID assigned prior to its deployment. These are just a few applications where location aware nodes are required.

Location discovery¹ in sensor networks is a challenging problem. Nodes need to determine their locations in a reliable manner while operating under strict constraints in computation, communication and energy resources. The design of a localization algorithm is influenced by many factors, including:

- **Scalability.** Scalability is one of the main factors that should be taken into consideration when designing a localization algorithm for sensor networks. The number of sensor nodes in the network may be on the order of hundreds or thousands. A location discovery algorithm that uses flooding to propagate position information will cause a scaling problem since the sheer number of sensor nodes makes such a global flooding undesirable. When thousands of nodes communicate with each

¹Also called, in the literature, the node localization problem.

other, broadcast storms may result in significant power consumption and possibly a network meltdown. A localization algorithm has to be distributed in order to scale well for large sensor networks.

- **Power Consumption.** Wireless sensors are usually equipped with a limited power source. Moreover, in some application scenarios, replenishment of power resources might be impossible. Therefore energy conservation is one of the major system design factors. A location discovery algorithm should be designed such that the communication overhead between nodes is as minimum as possible. It can also be designed by appropriately trading off accuracy with power efficiency.
- **Sensor Network Topology.** Deploying a high number of unattended sensor nodes, which are prone to frequent failures, make topology maintenance a challenging task. Due to the initial random deployment, a localization algorithm must possess self-organizing capabilities; hence; it should not rely on any infrastructure information. Moreover, a localization algorithm should not assume that any node in the network has previous knowledge of position information.
- **Accuracy.** A localization algorithm should determine the node's positions with acceptable accuracy. Many of the localization algorithms often assume that each sensor node will contain a global positioning system (GPS). Unfortunately, the straightforward solution of adding GPS to all the nodes in the network is not practical for several reasons [75]:

1. **Cost** – Sensor nodes are assumed to have low production cost and be dispens-

able. If we are envisioning a network of thousands, or tens of thousands of nodes, the production cost will dramatically increase.

2. **Line-of-Sight (LOS) conditions** – GPS requires line-of-sight signal reception from the GPS satellites. However, nodes may be deployed indoors, in the presence of dense vegetation, foliage, or GPS reception might be obstructed by climatic conditions.
3. **High power consumption**– Sensor nodes have very limited power. The power consumption of GPS will reduce the battery life of the sensor nodes thus reducing the effective lifetime of the entire network.
4. **Forming factor** – The sensor node may need to fit into a matchbox-sized module [56]. The required size may be smaller than even a cubic centimeter [71]. The size of GPS and its antenna increases the sensor node form factor.

As a conclusion, with ad-hoc deployment one cannot accurately predict or plan a-priori the location of each sensor node. Using GPS is not always a suitable solution. Based on these facts, we propose SALAM, a scalable GPS-free (*anchor-free*) localization algorithm that addresses the above design issues imposed by sensor networks while determining the position of sensor nodes with consistent error margin.

1.3 Proposed Solution

Clustering is a standard approach for achieving efficient and scalable performance in wireless sensor networks. Clustering facilitates the distribution of control over the network and, hence, enables locality of communication. Clustering nodes into groups saves energy and reduces network contention because nodes communicate their data over shorter distances to their respective cluster heads instead of network-wide flooding. Moreover, cluster-based protocols are robust to network partitioning and node failure.

In this dissertation, we present SALAM, an anchor-free cluster-based localization protocol that can determine the position of sensor nodes consistently with low error margins and without any infrastructure support. We assume that each node has the capability to estimate *ranges* (distances) to its corresponding neighbors, that are within its transmission range, with some error. The network is divided into *overlapping* multi-hop clusters each with its own cluster head node. Each cluster head is responsible for building a local relative map corresponding to its cluster using intra-cluster node's range measurements. We formulate an optimization model to minimize the cumulative intra-cluster errors that may affect the accuracy of the established relative coordinate system. The cluster head nodes collaboratively combine their local maps to obtain the global relative topology of the network. A global coordinate system can be built from the local maps available at each cluster head using simple matrix rotations, translations, and mirroring. We find the best order of transformations to minimize the inter-cluster error that may affect the global relative topology. In order to obtain absolute node positions, SALAM uses as few as three GPS-enabled anchor nodes.

In order for two cluster heads to perform these matrix transformations, there must be at least three *boundary nodes*² (i.e. the two clusters are overlapping with degree at least 3). We formulate the overlapping multi-hop clustering problem as an extension to the k -dominating set (KDS) problem. Then we propose the overlapped k -hop (OK) clustering algorithm, a randomized distributed algorithm, to solve the problem. After the termination of the clustering process, each node is either a cluster head or within k hops from *at least one* cluster head, where k , the *cluster radius*, is a parameter in the algorithm. After that each node discovers its neighbors that are within its transmission range and estimates their ranges and fuses the range measurements to the cluster head node.

A problem that occurs here is the error accumulated in the node position as it becomes multi-hop away from the cluster head node. One of the contributions of this dissertation is to show how the error accumulates, as the node becomes k -hop away from the cluster head node, and what factors affect this error accumulation. We also propose some heuristics to reduce this error.

A major motivation for our approach is that we believe that locally centralized algorithms scale well with increased network size and are robust to network partitioning and node failure. Yet, they can achieve acceptable accuracy compared to a centralized approach. A locally centralized algorithm should be a good compromise between accuracy, communication overhead.

²A *boundary node* is a node that belongs to more than one cluster.

1.4 Contributions

The contributions of this dissertation are:

- Formulating the *overlapping multi-hop clustering* problem as an extension to the k -dominating set (KDS) problem.
- Designing and implementing a distributed randomized multi-hop clustering algorithm (OK) for organizing the sensors in a wireless sensor network into overlapping clusters. We analyze the effect of different parameters (cluster radius, network connectivity, cluster head probability) on the performance of the clustering algorithm in terms of communication overhead, node coverage, average overlapping degree, and average cluster size. We also develop a detailed analytical model for the overlapped multi-hop clusters problem and validate it by comparison with the simulation results.
- Analyzing the problem of multi-hop relative location estimation and different sources of error and developing heuristics to avoid these errors. We design and implement the Multi-hop Relative Location Estimation (MRLE) algorithm that uses these heuristics to estimate relative node's positions with low error margins. We study the effect of local coordinate system (LCS) on the accuracy of the estimated position and propose different heuristics to select the LCS.
- Analyzing the accuracy of the intra-cluster location discovery and capturing the impact of different parameters, such as cluster radius and connectivity on the accuracy of the estimated position. We also introduce a new metric, the CLIQUE factor, to

measure how close a graph to the complete graph (CLIQUE). We show that the CLIQUE factor is one of the major factors affecting accuracy.

- Designing a policy to trade accuracy for energy and/or computational power. The application layer can choose from a whole range of different options, to estimate the sensor node's positions with different accuracy while conserving battery power.
- Formulating the problem of *best order of transformations* between clusters as a spanning tree problem. We introduce a new data structure, the *overlapping graph* and propose different heuristics to assign weights to the edges of the overlapping graph in order to optimize a certain design goal.
- Analyzing the accuracy of the global (inter-cluster) location discovery and capture the impact of the overlapping degree between clusters on the accuracy of the estimated node's positions.
- Comparing the performance of SALAM with other ad-hoc localization techniques for wireless sensor networks.

We also relay lessons learned and identify opportunities for future research.

1.5 Dissertation Organization

In chapter 2, we present a taxonomy of localization algorithms in wireless sensor networks and survey current research in this field. Chapter 3 gives a brief overview of SALAM and the system model used. In chapter 4, we discuss the problem of overlapping multi-hop clustering and formulate the problem as an extension to the k -dominating set

problem. We also present the overlapped K -hop (OK) clustering algorithm and analyze its performance both analytically and via simulations. We address the problem of multi-hop relative location estimation in chapter 5, and discuss the major sources of error. We present the multi-hop relative location estimation algorithm (MRLE) and evaluate the accuracy of the intra-cluster estimated position through simulations. Chapter 6 discusses the global location discovery (GLD) where the cluster head nodes collaborate to obtain a global map of the network. We also analyze the overall accuracy of SALAM and compare it with other localization schemes. Finally, Chapter 7 concludes the dissertation and gives directions for extending the research work.

Chapter 2

A Taxonomy of Localization Schemes for Wireless Sensor Networks

Although a good survey of location systems can be found in [52], the survey focused more on infrastructure-based location systems. Location discovery in sensor networks have been an active research area for the past couple of years. Several localization algorithms have been proposed and implemented. In this chapter, we describe a taxonomy consisting of several distinct features of a localization algorithm. Then we present an overview of earlier and current research in this field by surveying several localization algorithms. This survey is by no means exhaustive. Location discovery have been an extremely active field and in constant evolution for the past couple of years.

2.1 Taxonomy Features

Location discovery algorithms may be classified according to several criteria, reflecting fundamental design and implementation choices. Those different criteria form a reasonable taxonomy for characterizing and evaluating location discovery algorithms. In this section, we try to summarize different design alternatives for location discovery algorithms in general and in wireless sensor networks in particular.

2.1.1 Anchor-based versus Anchor-free

Anchor-based algorithms operate on an ad-hoc network of sensor nodes where a small percentage of the nodes (*anchors*) are aware of their positions either through manual configuration or using GPS. Anchor nodes broadcast their locations information to their neighbors. The goal is to estimate the positions of as many unknown nodes as possible using anchor node information. Anchor-based algorithms usually produce an absolute location system where absolute node position is known, for example, latitude, longitude, and altitude. However, the accuracy of the estimated position is highly affected by the number of anchor nodes and their distribution in the sensor field [17]. Langendoen et al. [63] showed that with anchor density of 20%, we could have an accuracy of 25% of transmission range, which falls short from the required inaccuracy in many applications. Moreover, most of these algorithms suffer from scalability problem. Propagating anchor node location information through the network may lead to a network-wide flooding.

Anchor-free algorithms do not make any assumptions regarding node positions. In this case, instead of computing absolute node positions, the algorithm estimates relative positioning, in which the coordinate system is established by a reference group of nodes. In some cases knowing the relative positions of the nodes compared to each other is enough, for example, location-aided routing [84, 67]. Moreover, a relative coordinate system can be transformed to an absolute coordinate system if the coordinates of three separate non-colinear nodes are known in case of 2D (or four in case of 3D). Anchor-free schemes have the disadvantage that when the reference node moves, positions have to be recomputed for nodes that have not moved. This is considered a minor problem in sensor

networks where sensor nodes are usually assumed to be stationary.

2.1.2 Range Estimation Method

Ranging is the process of estimating node-to-node distances or angles. The recent research work by He et al. [42] divides the location discovery algorithms in sensor networks into two major categories: range-based algorithms and range-free algorithms. The former is defined by protocols that use absolute point-to-point range (distance or angle) estimates for estimating location. The later make no assumption about the availability or validity of such information.

The most popular methods for estimating the range between two nodes are:

- **Time-based methods.** Time-of-Arrival (ToA) or Time-Difference-of-Arrival (TDoA) methods record the signal transmission time and the signal arrival time or the difference of arrival times. The propagation time can be directly translated into distance, based on the known signal propagation speed. These methods can be applied to many different signals, such as RF, acoustic, infrared and ultrasound.
- **Angle-of-Arrival methods.** Angle-of-arrival (AoA) methods estimate the angle at which signals are received and use simple geometric relationships to calculate bearings to neighboring nodes with respect to node's own axis.
- **Received-Signal-Strength-Indicator (RSSI) methods.** Received-Signal-Strength-Indicator (RSSI) methods measure the power of the signal at the receiver. Based on the known transmission power, the effective propagation loss can be calculated.

Theoretical and empirical models are used to translate this loss into a distance estimate. This method has been used mainly with RF signals.

- **Network Connectivity methods.** Network connectivity methods can be used for range estimation if the cost of range-estimation hardware is expensive or if a sensor cannot receive signals from enough base stations (≥ 2 for AoA, ≥ 3 for ToA, TDoA, and RSSI). In this case, network connectivity can be exploited for range estimation. For example, the number of hops between two nodes can be used as an estimate of the range between these two nodes.

2.1.3 Range Combining Technique

Once a location discovery algorithm estimates ranges to other neighboring nodes, it tries to estimate node position using the estimated ranges. The most known techniques for combining ranges are:

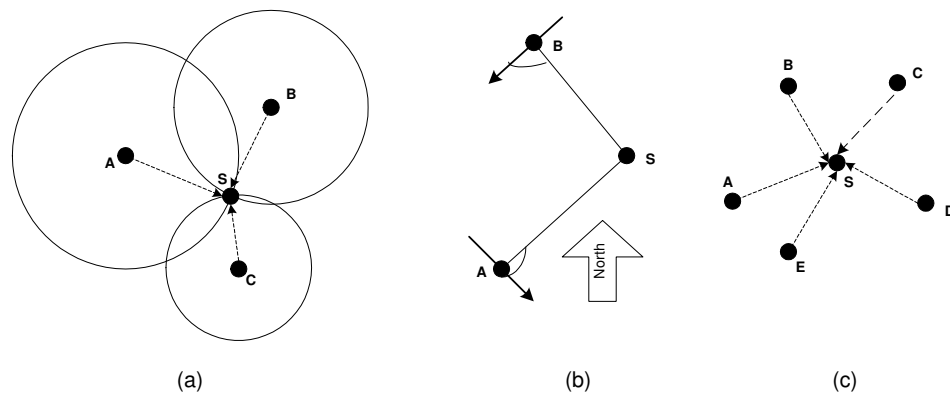


Figure 2.1: Range Combining Techniques: (a) Trilateration, (b) Triangulation, (c) Multilateration

- **Trilateration.** Trilateration locates a node by calculating the intersection of three

circles as shown in Figure 2.1a. If the ranges contain error, the intersection of the three circles may not be a single point.

- **Triangulation.** Triangulation is used when the angle of the node instead of the distance is estimated, as in AoA methods. The node positions are calculated in this case by using the trigonometry laws of sines and cosines. In this case, at least two ranges are required as shown in Figure 2.1b.
- **Multilateration.** In multilateration, the position is estimated from distances to three or more known nodes by minimizing the error between estimated position and actual position [30]. For example, in Figure 2.1c, we can use the following function to compute the location (x, y) of node S .

$$\min \sum_i (D_{S,i} - \hat{D}_{S,i})^2, \quad (2.1)$$

where $D_{S,i} = \sqrt{(x - x_i)^2 + (y - y_i)^2}$, $\hat{D}_{S,i}$ is the estimated range from S to i , $i = A, B, C, D, E$.

- **Proximity-based.** Proximity-based is usually used when no range information is available. In a simple proximity based approach, position of a node is taken as the centroid of positions of connected anchor nodes. An anchor node is considered connected to a node if the percentage of messages received from the anchor node in a time interval t exceeds a certain threshold.

2.1.4 Computational Model

There are different possibilities how to construct the localization algorithm and how to divide the computation between nodes. A location discovery algorithm can be categorized under one of the following computational models:

- **Centralized.** In the centralized model, all the range measurements are collected to a central base station where the computation takes place and the results are forwarded back to the nodes.
- **Locally Centralized.** Locally centralized (*localized*) algorithms are distributed algorithms that achieve a global goal by communicating with nodes in some neighborhood only. For example, the sensor network can be divided into local clusters, where each cluster has a head. All the range measurements in a certain cluster are forwarded to the cluster head where computation takes place.
- **Fully Distributed.** In the fully distributed, computation takes place at every node. In other words, the cluster size is one. Each node is responsible for estimating its own position.

2.1.5 Accuracy

A location discovery algorithm should estimate sensor position accurately. Accuracy is usually measured as percentage of sensor transmission range. Accuracy usually depends on range measurement errors. Range measurements with less error will lead to more accurate position estimates. Moreover, in anchor-based algorithms, accuracy is affected

by the percentage and placement of anchor nodes in the network as well as the placement error.

2.1.6 Communication Power

Wireless sensors are usually equipped with a limited power source. Therefore energy conservation is one of the major system design factors. A sensor node spends maximum energy in data communication. This involves both data transmission and reception. The current generation of sensor platforms uses about $2 \mu\text{J}$ per bit of data transmitted [2]. Usually sensor nodes communicate over a shared medium, and a high density of nodes, coupled with a high messaging complexity, leads to a high collision rate and ultimately to lower throughput and higher power consumption. Therefore, a location discovery algorithm should minimize the amount of node-to-node communication. Data aggregation techniques can be used to conserve communication bandwidth.

2.1.7 Computation Power

The processor is the second main source of draining battery life. Current small batteries provide about 100mAh of capacity, this can power a small Amtel processor for 3.5 hours (if no power management techniques would be applied) [2]. Energy expenditure in data processing is much less compared to data communication. The example described in [71], effectively illustrates this disparity. Assuming Rayleigh fading and fourth power distance loss, the energy cost of transmitting 1 KB a distance of 100 m is approximately the same as that for executing 3 million instructions by a 100 million instructions per second (MIPS)

processor. Hence, a location discovery algorithm should use local data processing in order to minimize communication power.

2.1.8 Scalability

Scalability is one of the main factors that should be taken into consideration when designing a protocol or an algorithm for sensor networks. The number of sensor nodes in the network may be on the order of hundreds or thousands. Depending on the application, the number may reach an extreme value of millions [2]. For example, a location discovery algorithm should not use flooding to exchange information with other nodes. The sheer number of sensor nodes makes such a global flooding undesirable. A cluster-based approach would work better. Moreover, location discovery systems should not require large tables, which do not fit in the sensor node limited memory.

2.1.9 Capital Cost

Capital costs include factors such as the price per sensor node or extra hardware required for location determination. For example, a simple civilian GPS receiver costs around \$100. This increases the cost of the sensor node significantly making it impractical to develop.

2.1.10 Limitations

By limitations we mean the situations in which the location discovery algorithm fails to position the nodes with acceptable accuracy. Some algorithms may fail to work in-

doors or in the presence of dense vegetation, due to the range estimation technique used. Irregular topology shape is another factor affecting the performance of a localization algorithm. A positioning algorithm depending on the network connectivity may fail in case of anisotropic topology, as shown in Figure ??.

Another important factor is the network dynamicity. Sensor nodes are very prone to failure due to lack of power, physical damage, or environmental interference. This may cause the network to be disconnected. A positioning algorithm may fail if the network becomes disconnected or even if the degree of the node decreases under a certain threshold. Mobility is another factor that may affect the performance of a localization algorithm. In sensor networks since most of the applications assume that the network is stationary, mobility is not considered a major design factor.

2.2 An Overview of Location Discovery Algorithms

Although, a very good survey of location systems is provided in [52], very few systems are actually ad-hoc. Node localization has been the topic of active research and many systems have made their appearance in the past few years. In this section we consider several recent localization algorithms for sensor networks. Although, some forms of ad-hoc localization also exist in the domain of mobile robotics [54, 73], we focus more on the ad-hoc localization problem investigated in the context of sensor networks. One main difference between mobile robots and sensor networks is that mobile robots have additional odometric measurements that can help with estimating the initial robot positions [73], something that is not available in sensor networks. Furthermore, localization studies in the sensor

network community also consider scalability communication and power consumption issues that are not studied by the robotics community. The localization algorithm overviews below are intended to emphasize key design issues and how they relate to the taxonomy described in section 2.

2.2.1 Hop-TERRAIN

Hop-TERRAIN [74] is a range-based anchor-based distributed algorithm. The algorithm consists of two phases: the start-up phase and the refinement phase. In the start-up phase, anchor node location information is propagated across the network. When anchors become aware of other anchor node locations, they use this information to estimate the average hop length in their vicinity and broadcast it back into the network. Nodes with unknown locations also note the shortest hop distance to each of the anchor nodes and multiply it with the broadcasted average hop length to estimate the approximate range between the node and each anchor. These computed ranges are then used together with the anchor nodes' known positions to perform a triangulation and get an initial estimated nodes' position. The triangulation consists of solving a system of linear equations by means of a least squares algorithm. These initial estimates are not expected to be very accurate, but are useful as rough approximations.

The start-up phase algorithm is run once at the beginning of the positioning algorithm. The Refinement phase is run iteratively afterwards to improve upon and refine position estimates generated by the start-up phase algorithm. Simulation studies have shown that these technologies are independent of ranging technologies and can deliver

localization accuracy within one third of the communication range.

2.2.2 APS Algorithm

The APS algorithm [68] belongs to the class of anchor-based range-free algorithms. Anchor nodes (*beacons*) flood their location to all nodes in the network using some propagation method. When anchors become aware of other anchor node locations, they use this information to estimate the average hop length in their vicinity and broadcast it back into the network. Nodes with unknown locations also note the shortest hop distance to each of the anchor nodes and multiply it with the broadcasted average hop length to get an approximate distance to each of the anchor nodes. With this information nodes perform a multilateration to get an initial estimate of their locations.

The paper discussed three methods of hop-to-hop distance propagation:

1. **DV-Hop Propagation Method.** Each unknown node records the position and minimum number of hops to at least three beacons. Whenever a beacon, b_i , infers the position of other beacons, it computes the average hop distance using Eq.(Eq:APShopDist), and floods this average hop distance into the network. Average hop distance =
$$\frac{\sum \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\sum h_j}, i \neq j$$

Where h_j is the number of hops between beacon b_i and b_j . Each unknown node then uses the average hop distance to convert hop counts to distances, and performs a triangulation to three or more distant beacons to estimate its own position.

2. **DV-Distance Propagation Method.** This method is similar with the previous one except that distance between neighboring nodes is measured using radio signal

strength and is propagated in meters rather than in hops.

3. **Euclidean Propagation Method.** In this method, the true Euclidean distance to the beacon is propagated. An unknown node needs to know an estimate of distance to at least two neighbors, which have estimates for the distance to a beacon. Then using simple geometry, the unknown node can estimate its distance to the beacon node.

2.2.3 GPS-less

The GPS-less [16] system is a distributed range-free anchor-based technique. It uses connectivity between nodes in order to estimate node positions. The system employs a grid of beacon nodes, powerful (compared to the nodes) base stations, with known locations; each unknown node sets its position to the centroid of the beacon locations it is connected to. Besides relying on infrastructure support, the accuracy of estimated position depends highly on the density of the beacons. The reported position accuracy is about one-third of the separation distance between beacons.

2.2.4 Convex Position Estimation

Convex position algorithm [37] belongs to the class of centralized range-free anchor-based localization algorithms. The algorithm uses the connectivity between nodes to formulate a set of geometric constraints and solve it using convex optimization. If one node can communicate with another, a proximity constraint exists between them. For example, if particular RF system can transmit 20m and two nodes are in communication, their

separation must be less than 20m. These constraints restrict the feasible set of unknown node positions. Formally, the network is a graph with n nodes at the vertices and with bidirectional communication constraints as the edges. Positions of the first m nodes, *anchor nodes*, are known $(x_1, y_1 \dots x_m, y_m)$ and the remaining $n - m$ positions are unknown. The problem is then to find $(x_{m+1}, y_{m+1} \dots x_n, y_n)$ such that the proximity constraints are satisfied. The algorithm is based on semi-definite programming and requires rigorous computation so it is not always suitable for sensor networks. The resulting accuracy depends on the fraction of anchor nodes. A serious drawback is that convex optimization is performed by a single, centralized node; hence; it is not suitable for many ad-hoc setups.

2.2.5 Iterative Multilateration

Iterative multilateration [75] is used in the AHLoS project. The algorithm is fully distributed and can run on each individual node in the network. An unknown node u that is connected to at least three beacons can estimate their position by solving the following system of equations:

$$d_{iu}^2 = (x_i - x_u)^2 + (y_i - y_u)^2 \forall u \in U \text{ and } i \in B_u \quad (2.2)$$

Where B_u is the set of all beacon neighbors of u . The resulting system of equations can be linearized by rearranging terms, and subtracting the last equation from the rest to obtain the following equation:

$$a_i x_u + b_i y_u = c_i, \text{ where } a_i, b_i, \text{ and } c_i \text{ are constants.} \quad (2.3)$$

This system of equations can be solved using the matrix solution for minimum mean square estimate (MMSE) [75].

Once a node estimates its position it becomes a beacon and can assist other unknown nodes in estimating their positions by propagating its own location estimate through the network. This process iterates to estimate the locations of as many nodes as possible.

Iterative multilateration requires high fraction of beacon nodes. It is sensitive to beacon densities and can easily get stuck in places where beacon densities are sparse. Another drawback of iterative multilateration is the error accumulation that results from the use of unknown nodes that estimate their positions as beacons.

2.2.6 Collaborative Multilateration

Collaborative multilateration [77, 76], also known as The n -Hop Multilateration can be used in case an unknown node does not have at least three neighboring beacons and therefore cannot estimate its location using iterative multilateration. Collaborative multilateration is a multilateration that spans over multiple hops. This enables nodes that are not directly connected to beacon nodes to collaborate with other intermediate nodes with unknown locations situated between themselves and the beacons to jointly estimate their locations.

Collaborative multilateration consists of three main phases and a post-processing phase. In the first phase, the nodes self-organize into groups, collaborative subtrees so that nodes with unknown positions are over-constrained and can have only one possible solution. During the second phase, the nodes use simple geometric relationships between measured distances and known beacon locations to obtain a set of initial position estimates. In the third phase, node locations are computed by setting up a global non-linear

optimization problem and solving it using a Kalman filter. The solution is presented in two computation models, centralized and a fully distributed approximation of the centralized model.

2.2.7 GPS-free

In their GPS-free system [21], Capkun et al introduced the Self-Positioning Algorithm (SPA) that enables nodes in a MANET to find their positions in the network using range measurements between the nodes. The TOA method is used to obtain range between two mobile devices. Each node in the network builds its own local coordinate system by assuming itself as the origin of this coordinate system, and selecting two non-collinear one-hop neighbors to form axes. Then the positions of one-hop neighbors are computed accordingly. The local coordinate systems at each node can have different directions so another phase is required to map all the local coordinate systems of the nodes to the network coordinate system. This is done through simple matrix rotations and may be mirroring. To compute the network coordinate system, a subset of nodes is chosen (Location Reference Group) such that it is stable and less likely to disappear from the network. Then the network coordinate center is chosen to be the geometrical center of the location reference group and the direction of the network coordinate system is the mean value of all directions of the local coordinate systems of the nodes belonging to location reference group. As the nodes move, the location reference group is periodically updated using expensive message broadcast. Moreover, the network center and direction are recomputed using expensive message broadcast. The work was focusing on the network mobility

and how this affects the localization accuracy. Power consumption at each node was not considered as a major problem.

2.2.8 MDS-MAP

MDS-MAP [80] is a localization method based on multidimensional scaling (MDS). It determines the positions of nodes given only basic information that is likely to be already available, namely, which nodes are within communications range of which others. If the distances between neighboring nodes can be measured, that information can be easily incorporated into the method. MDS-MAP is able to generate relative maps that represent the relative positions of nodes when there are no "anchor" nodes that have known absolute coordinates. When the positions of a sufficient number of anchor nodes are known, e.g., 3 anchors for 2-D localization and 4 anchors for 3-D, MDS-MAP then determines the absolute coordinates of all nodes in the network. MDSMAP often outperforms previous methods when nodes are positioned relatively uniformly in space, especially when the number of anchors is low. MDS-MAP uses the distance or connectivity information between all nodes at the same time, whereas previous triangulation-based methods localize one unknown node at a time and only use the information between the unknown and anchor nodes.

However, like many existing methods, MDS-MAP does not work well on irregularly-shaped networks, where the shortest path distance between two nodes does not correlate well with their true Euclidean distance.

2.2.9 Improved MDS-MAP

In [79], an enhanced version of MDS-MAP is proposed that works well on both uniform and irregular networks. The main idea is to compute a local map using MDS for each node consisting only of nearby nodes, and then to merge these local-area maps together to form a global map. The new technique is called MDS-MAP(P), which stands for MDS-MAP using patches of relative maps. This approach avoids using shortest path distances between far away nodes, and the smaller local maps, constructed using local information, are usually quite good. An optional refinement step using least-square minimization may be used to refine the relative maps computed by MDS. MDS is often good at finding the right general layout of the network, but not the precise locations of nodes. That makes the MDS solution a good starting point for the local optimization done in the refinement step. The refinement improves solution quality but is much more expensive than MDS. MDS computes analytical solutions in $O(n^3)$, where n is the number of nodes. Thus, the refinement provides a trade-off between solution quality and computational cost.

Chapter 3

SALAM Overview

3.1 Introduction

Node localization has been the topic of active research and a number of systems have been proposed over the past few years. Many of those systems fall into one of three classes or a combination of them. The first class includes *range-free* algorithms, which assume that there is no distance/angle information available at each node [57, 37, 68, 16]. Hence, they try to use the basic proximity information available at each node, i.e. which nodes are nearby. In general, *range-free* techniques provide the lowest level of accuracy among the three classes. The second class includes *anchor-based* algorithms [30, 69, 74, 76, 75], where nodes know their positions usually using GPS. Most of the approaches in this class require a high percentage of anchor nodes in order to reach an acceptable accuracy. Moreover, propagating anchor node location information through the network may lead to a network-wide flooding [57, 30, 42, 69, 76, 68, 75]. Besides, the inclusion of a GPS receiver on each node is not practical.

The third class of localization systems are *anchor-free* [80, 79, 21]. In this case,

instead of computing absolute node positions, the algorithm estimates nodes' positions relative to a coordinate system established by a reference group of nodes. A relative coordinate system can still be transformed to absolute coordinate system by using only three anchor nodes in case of 2-D (or four anchors in case of 3-D). Algorithms in this class can be range-free or range-based. The multi-dimensional scaling (MDS) [80] is an example of *range-free anchor-free* algorithms. Each node computes a local map for nodes that are within 2 hops using mainly node connectivity. Then all the nodes in the network communicate with each other to merge these local maps together to form a global map. The Self-Positioning Algorithm (SPA) [21] is an example of *range-based anchor-free* methods. Each node also builds its own local coordinate system, estimates the positions of *one-hop* neighbors using triangulation and broadcasts this information to all the nodes in the network to build a global network coordinate system.

SALAM belongs to the class of *anchor-free range-based* localization algorithms. However, SALAM is different. In SALAM, instead of forming a local coordinate system at each node like SPA and MDS, we build a cluster-wide coordinate system only at each cluster head node. In this case we gain the following benefits: (1) since the cluster head node has more information about the intra-node distances, we can use non-linear optimization techniques to estimate the node's positions more accurately; (2) the communication overhead to build global network topology is reduced since only cluster head nodes communicate with each other. While other anchor-free mechanisms consider nodes that 1 or 2 hops away, we estimate the position of nodes that are within k -hops from the cluster head node, where k , the *cluster radius*, is a parameter in our algorithm. SPA and MDS can be viewed as a special case from SALAM where $k = 1, 2$ respectively. In large

sensor networks, SPA and MDS generate a large number of cluster heads and eventually lead to the same problem as if there is no clustering. Another important difference between SALAM and MDS is that MDS-based methods are often good at finding the general layout of the network, but not the precise locations of nodes; hence, the resulting topology from MDS-based technique may require scaling. In chapter 6, we compare between MDS-based techniques, and SALAM and show that SALAM achieves a higher accuracy with less computational overhead.

SALAM consists of three phases: (i) network bootstrapping and cluster formation, (ii) local location discovery (LLD), and (iii) global location discovery (GLD). In the remainder of this chapter, we present an overview of the different phases. The detailed analysis of each phase, along with validation and performance evaluation, is presented in the next three chapters. We start by describing the considered system model in the next section.

3.2 System Model

We consider a typical sensor network which consists of sensor nodes that are scattered in an area of interest to detect and possibly track events/targets in this area. All nodes are alike and each node is assigned a unique id prior to deployment. Each sensor node is equipped with data processing and communication capabilities. The nodes are location-unaware, i.e. not equipped with GPS. There are neither base stations nor infrastructure support to coordinate the activities of subsets of nodes. Therefore, all the nodes have to collectively make decisions. We assume that the nodes are static. This assumption about

node mobility is typical for sensor networks. All sensors transmit at the same power level and hence have the same transmission range (T_r). Sensors are also capable of long -haul communications, however, this is mainly used by cluster heads to communicate with each other to build the global network topology during the GLD phase. We also assume that nodes have timers, but we do not require time synchronization across the nodes. Timers are used for tasks such as timing out of a node when waiting on a condition.

All communication is over a single shared wireless channel. A wireless link can be established between a pair of nodes only if they are within the transmission range of each other. We only considers bidirectional links. It is assumed the MAC layer will mask unidirectional links and pass bidirectional links to SALAM. Two nodes that have a wireless link will, henceforth, be said to be 1-hop away from each other. They are also said to be immediate neighbors. Nodes can identify neighbors using beacons. Each sensor node is also capable of estimating the distance to neighboring nodes using any range estimation technique as discussed in section 2.1.2, however, SALAM is independent of the ranging technology used.

3.3 Phase I: Range Estimation and Cluster Formation

This phase usually starts during network bootstrapping after the sensor nodes are deployed in the sensor field. The main tasks performed during this phase are: range estimation and cluster formation. We are interested in organizing the sensor network into multi-hop *overlapping* clusters as shown in Fig. 3.1. At the end of the clustering process, each node is either a cluster head or within k hops from *at least one* cluster head node, where k , the

cluster radius, is a parameter in our algorithm. We require overlapping between clusters with at least three nodes in order to be able to merge local cluster maps later during the GLD phase using matrix transformations. Also by the end of this phase, each cluster head should have all inter-node distances as reported by members of the clusters.

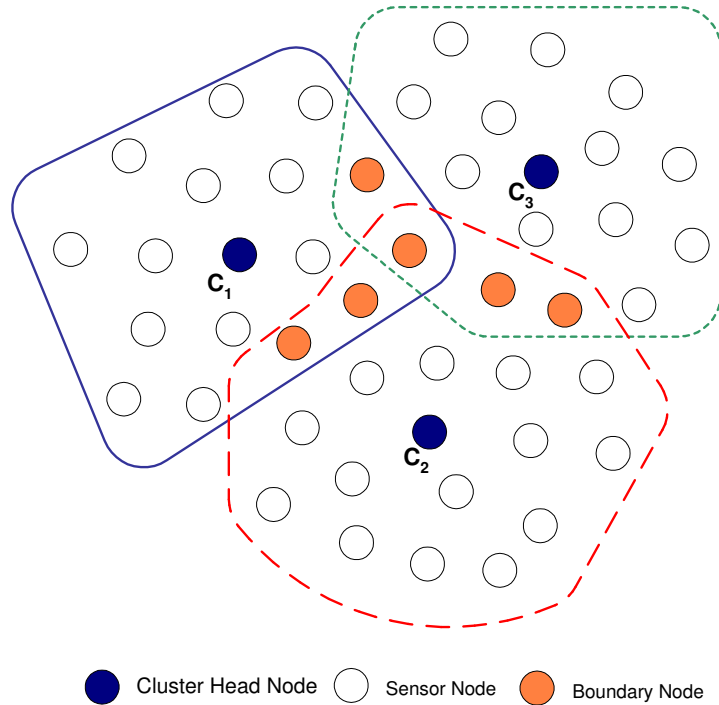


Figure 3.1: Organizing the sensor network into overlapping multi-hop clusters

In the last few years, there have been few clustering algorithms designed for sensor networks [88, 9, 49, 36, 47, 45]. None of those algorithms construct overlapping clusters. To the best of our knowledge, this is the first document to discuss the problem of overlapping multi-hop clustering. We formulate the overlapping k -hop clustering problem as an extension to the k -dominating set (KDS) problem. Then we propose the overlapped k -hop (OK) clustering algorithm, a randomized multi-hop distributed algorithm to solve the problem. The nodes randomly elect themselves as cluster heads (CH) with some

probability p . The cluster head probability (p) is another parameter in the algorithm that can be tuned to control the number of overlapping clusters in the network. Each cluster head node broadcasts an advertisement message asking neighbors to join the cluster. Each non-CH node then joins all the clusters it hear from and reports measured distance to neighbors to the cluster head node. We shall refer to the set of nodes that belong to more than one cluster as *boundary nodes*. Boundary nodes are essential for the global localization phase as we will discuss later.

3.4 Phase II: Local Location Discovery (LLD)

The goal of the LLD phase is to build a local map at each cluster head using the range measurements reported by the members of the cluster in the previous phase. The idea is to form a local coordinate system (LCS) at each cluster head nodes as shown in Fig 3.2. The cluster head node assumes that it is located at the origin of the cluster LCS and selects one neighbor node to form the x -axis, we call this neighbor node the first reference point R_1 . Then a second reference point (R_2) is selected such that the y -axis is perpendicular to the x -axis in the direction of R_2 . We show that the LCS affects the accuracy of the estimated position dramatically and we propose different heuristics to select the LCS and compare between those heuristics.

After selecting the LCS, we introduce the Multi-hop Relative Location Estimation (MRLE) algorithm. MRLE uses a combination of triangulation and trilateration techniques to find an initial position estimate (P_0) for the nodes located within the cluster using the received measurements of inter-node distances.

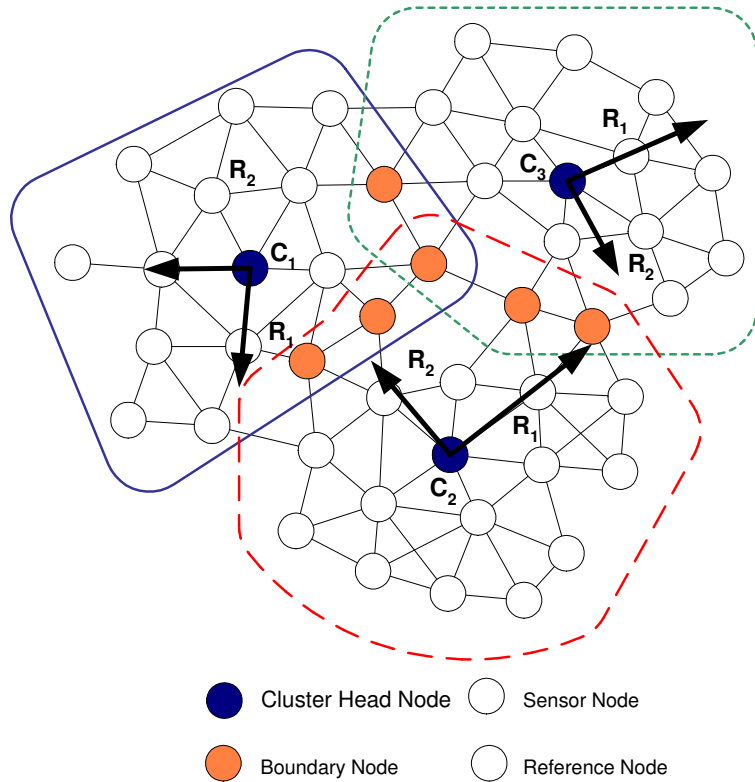


Figure 3.2: Building local coordinate system (LCS) within each cluster

Finally, we introduce an optional refinement step, where we iteratively improve the initial position estimate by formulating a least-squares metric and solving it using non-linear optimization techniques. Compared with the MREL algorithm, the refitment step is much more expensive, in terms of computation power. Hence, it is optional if we are seeking higher accuracy. We also introduce a new metric, the CLIQUE factor (CF). The CLIQUE factor of a cluster measures how close the subgraph induced by cluster to a complete graph. The simulation results show that the CF is the major factor affecting accuracy regardless of cluster size. The CLIQUE factor is a function of the node transmission range; hence we can use it to trade accuracy for transmission power. We show that SALAM is an adaptive fidelity algorithm that gives a sensor network engineer the

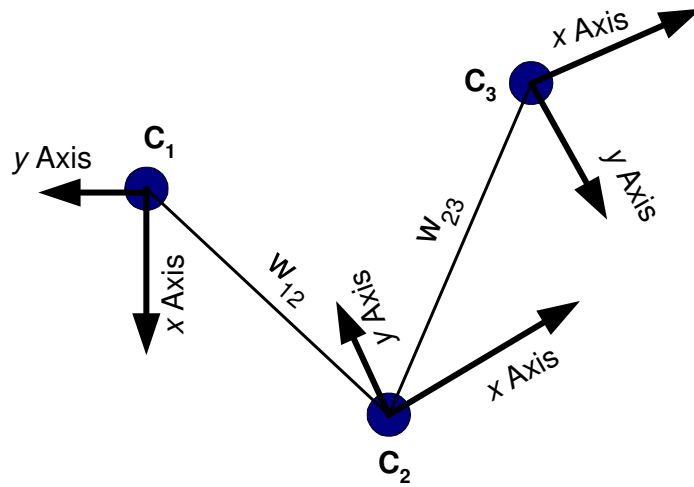


Figure 3.3: Global network map can be obtained from local maps using simple matrix transformations

ability to trade accuracy for either transmission power or computational power or both.

3.5 Phase III: Global Location Discovery (GLD)

In the GLD phase, the cluster head nodes collaborate to obtain a global map of the network. This phase can be optional if a global view of the network is not needed. After forming a cluster-level map during the LLD phase, each cluster head has its own local coordinate system (LCS). The axes of the LCS usually have different directions as shown in Fig. 3.3. Two local maps have the same direction if their x -axes are pointing in the same direction and similarly for the y -axes (and z -axes in case of 3-D). A global coordinate system can be built from the local maps available at each cluster head using simple matrix rotations, translations, and mirroring.

To be able to perform such transformation, there must be at least three *boundary*

nodes that belong to both clusters. We describe how to adjust the directions of the local maps of the cluster head nodes to obtain the global topology of the network using boundary nodes and show how the number of boundary nodes (*overlapping degree*) between two clusters affect the accuracy of the transformation from one coordinate system to another.

Finally, we introduce a new problem, the best order of transformations between clusters. To formulate the problem, we define a new data structure, the *overlapping graph*. Each vertex in the overlapping graph represents one cluster in the network where an edge between two vertices implies that there is at least three boundary nodes (i.e. a minimum overlapping degree of 3) between the two corresponding clusters. Hence if an edge exists between two vertices u and v , this means that we can transform from the local coordinate system corresponding to u to that of v and vice versa.

Given an overlapping graph of m cluster head nodes, in order to find the global map of the network, we need to perform $m - 1$ transformations in order to merge m local coordinate systems. This is equivalent to finding a spanning tree (ST) for the overlapping graph. There are many spanning trees that can link the cluster head nodes together. Each spanning tree corresponds to a different order of transformations between the local coordinate systems. Each order of transformations will result in a different accuracy, and different communication overhead per node. We will refer to the problem of finding the spanning tree, that satisfies a certain design goal, as *the best order of transformations* problem. We propose different heuristics to assign weights to the edges of the overlapping graph in order to optimize a certain design objective. We also propose two approaches to construct the spanning tree of the overlapping graph. In the results section, we show how

the spanning tree highly affects both accuracy and communication overhead of the GLD phase.

Chapter 4

The Overlapped K-hop (OK) Clustering Algorithm

4.1 Introduction

Clustering is a standard approach for achieving efficient and scalable performance in wireless sensor networks. Clustering facilitates the distribution of control over the network and, hence, enables locality of communication. Clustering nodes into groups saves energy and reduces network contention because nodes communicate their data over shorter distances to their respective cluster heads. The cluster heads forward the aggregated information to the base station. Only the cluster heads need to communicate far distances to the base station; this burden can be alleviated further by hierarchical clustering, i.e., by applying clustering recursively over the cluster heads of a lower level.

Many clustering protocols have been investigated as either stand alone protocols [8, 38, 10, 12, 35, 66, 11, 25, 6, 31, 62, 9, 7, 88, 49, 36] or as a side effect of other protocol operations, e.g., in the context of routing protocols [59, 65, 49], or in topology management protocols [86, 26, 23]. The majority of those protocols construct clusters where every node in the network is no more than 1 hop away from a cluster head [8, 38, 12, 10,

35, 88, 36, 49]. We call these *single hop* (1-hop) clusters. In large networks this approach may generate a large number of cluster heads and eventually lead to the same problem as if there is no clustering. Few papers have addressed the problem of *multi-hop* (k -hop) clustering [7, 9]. These algorithms are mostly heuristic in nature and aim at generating the minimum number of disjoint clusters such that any node in any cluster is at most k hops away from only one cluster head. The proposed OK clustering algorithm belongs to the multi-hop category.

In the last few years, there have been few clustering algorithms designed for sensor networks [88, 9, 49, 36, 47, 45]. Most of those algorithms aim at generating the minimum number of disjoint clusters that maximize the network lifetime. The algorithms discussed in [49, 88, 9] are randomized where the sensors elect themselves as cluster heads with some probability p and broadcast their decisions to neighbor nodes. The remaining sensors join the cluster of the cluster head that requires minimum communication energy. The proposed OK clustering protocol belongs to the class of randomized algorithms. Both the HEED algorithm [88] and LEACH algorithm [49] form single-hop non-overlapping clusters with the objective of prolonging network lifetime. In [9], the authors proposed a LEACH-like randomized multi-hop clustering algorithm for organizing the sensors in a hierarchy of clusters with an objective of minimizing the energy spent in communicating the information to the processing center. None of the above algorithms construct *overlapping* clusters.

In this chapter, we propose a fast, randomized, distributed multi-hop clustering algorithm (OK) for organizing the sensors in a wireless sensor network in *overlapping* clusters. After the termination of the clustering process, each node is either a cluster head

or within k hops from *at least one* cluster head, where k (*cluster radius*) is a parameter in the algorithm. To the best of our knowledge, this is the first document to discuss the problem of overlapping multi-hop clustering. OK operates in quasi-stationary networks where nodes are location-unaware and have equal significance. The protocol incurs low overhead in terms of processing cycles and messages exchanged. OK was designed with the following goals:

1. Is completely distributed (i.e. each node independently makes its decisions based on local information and without any centralized control).
2. Is scalable in terms of processing time (i.e. the clustering process terminates in a constant time independent of network size) and in terms of communication overhead (the number of control messages transmitted by node is independent of network size).
3. Does not make any assumptions about the location of the nodes.
4. Is asynchronous (Due to the large number of nodes involved, it is desirable to let the nodes operate asynchronously. OK does not assume any kind of clock synchronization between nodes, hence, The clock synchronization overhead is avoided, providing additional processing savings).
5. Is energy efficient in terms of processing complexity and message exchange (control overhead is linear in the number of nodes).
6. Is efficient in terms of memory used by the data structures required to implement the algorithm.

7. Chooses cluster heads that are well distributed over the sensor field.
8. Allows multi-hop clusters to be formed.
9. Ensures overlapped clusters with some average overlapping degree.

To the best of our knowledge, the proposed algorithm is the first algorithm to address the above goals in an integrated manner. We formulate the overlapping k -hop clustering problem as an extension to the k -dominating set problem [48]. Then we propose OK, a randomized multi-hop distributed algorithm to solve the problem. The nodes randomly elect themselves as cluster heads with some probability p . The cluster head probability (p) is another parameter in the algorithm that can be tuned to control the number of overlapping clusters in the network. The clustering process terminates in $O(1)$ iterations, independent of the network diameter. It does not depend on the network topology or size. We also analyze the effect of different parameters (e.g. node density, network connectivity) on the performance of the clustering algorithm in terms of communication overhead, node coverage, and average cluster size. The results show that although we have overlapped clusters, the OK clustering algorithm still produces approximately equal-sized clusters, which is a desirable property because it enables an even distribution of control between cluster head nodes.

The chapter is organized as follows. section II briefly surveys related work. Section III states the overlapping k -hop problem. Section IV presents the OK protocol architecture and proves that it satisfies its design goals. Section V shows the performance of OK via simulations and in section VI, we provide analytical models for the results. We study the complexity and correctness of the proposed protocol in section VII.

4.2 Related Work

In the last few years, many algorithms have been proposed for clustering in wireless ad-hoc networks [88, 49, 36, 9, 7, 47, 45, 8, 38, 10, 12, 35, 66, 11, 25, 6, 31, 62]. Clustering algorithms can be classified as either deterministic or randomized. Deterministic algorithms [11, 12, 25, 8, 38, 7, 65, 70] use weights associated with nodes to elect cluster heads. These weights can be calculated based on number of neighbors (*node degree*) [11, 12], node id [8, 38, 7], residual energy, and mobility rate [25]. Each node broadcasts the calculated weight. Then a node is elected as a cluster head if it is the highest weight among its neighboring nodes. In randomized clustering algorithms, the nodes elect themselves as cluster heads with some probability p and broadcast their decisions to neighbor nodes [49, 88, 10, 9, 13]. The remaining nodes join the cluster of the cluster head that requires minimum communication energy. The probability p is an important parameter in a randomized algorithm. It can be a function of node residual energy [49] or hybrid of residual energy and a secondary parameter [88]. In [9], the authors obtain analytically the optimal value for p that minimizes the energy spent in communication. In OK, the probability p is tuned to control the number of overlapping clusters in the network.

The Distributed Clustering Algorithm (DCA) [12] elects the node that has the highest node degree among its 1-hop neighbors as the cluster head. The DCA algorithm is suitable for networks in which nodes are static or moving at a very low speed. The Distributed and Mobility-adaptive Clustering Algorithm (DMAC) [11] modifies the DCA algorithm to allow node mobility during or after the cluster set-up phase. The Weighted

Clustering Algorithm (WCA) [25] calculates the weight based on the number of neighbors, transmission power, battery-life and mobility rate of the node. The algorithm also restricts the number of nodes in a cluster so that the performance of the MAC protocol is not degraded. In the Linked Cluster Algorithm (LCA) [8], a node becomes the cluster head if it has the highest identity among all nodes within one hop of itself or among all nodes within one hop of one of its neighbors. The LCA algorithm was revised [38] to decrease the number of cluster heads produced in the original LCA. In this revised version of LCA (LCA2), the algorithm elects as a cluster head the node with the lowest id among all nodes that are neither a cluster head nor are within 1-hop of the already chosen cluster heads. Both LCA and LCA2 heuristics were developed to be used with small networks of less than 100 nodes. As the number of nodes in the network grows larger, LCA/LCA2 will impose greater delays between node transmissions in the TDMA communication scheme and may be unacceptable.

Many of these clustering algorithms [8, 38, 25, 11, 65] are specifically designed with an objective of generating stable clusters in environments with mobile nodes. But in a typical wireless sensor network, the sensors' locations are fixed and the instability of clusters due to mobility of sensors is not an issue. However, the network is still dynamic because of node failure or adding new nodes. Moreover, the clustering time complexity in some protocols [12, 25, 10, 65, 70] is dependent on the network diameter. Most of these algorithms have a time complexity of $O(n)$, where n is the total number of nodes in the network. This makes them less suitable for sensor networks that have a large number of sensors. Unlike those protocols, OK terminates in a constant number of iterations.

Some clustering algorithms make assumptions about node capabilities, e.g., location-

awareness or clock synchronization. In [86, 87, 23, 26], the geographic location of each node is assumed to be available based on a positioning system such as GPS or through broadcast messages and routing updates [SPAN]. This is again not a reasonable assumption in case of low-cost low-power sensor networks. The clustering algorithm proposed in [Chiasserini02] assumes that each node is aware of the whole network topology, which is usually impossible for wireless sensor networks with a large number of nodes. Some algorithms [8, 38, 49, 25, 12, 11] require time synchronization among the nodes, which makes them suitable only for networks with a small number of sensors.

The majority of clustering algorithms construct clusters where every node in the network is no more than 1 hop away from a cluster head [8, 38, 12, 25, 11, 10, 35, 88, 36, 49]. We call these single hop (1-hop) clusters. For example, the HEED [88] algorithm forms single-hop non-overlapping clusters with the objective of prolonging network lifetime. Cluster heads are randomly selected according to a hybrid of their residual energy and a secondary parameter, such as node proximity to its neighbors or node degree. A careful selection of the secondary clustering parameter can balance load among cluster heads. HEED performance was analyzed assuming synchronized nodes. However, the authors showed that unsynchronized nodes can still execute HEED independently, but cluster quality will be affected. In [36], the authors present a clustering algorithm (FLOC) that produces non-overlapping and approximately equal-sized clusters. The clustering is such that all nodes within one hop from a cluster head belongs to its cluster, and no node m hops away from the cluster head may belong to its cluster. In [47, 45] the clustering algorithm assumes gateway (*master*) nodes are already known and the objective is to perform load balancing between different clusters by changing cluster radius. In large networks

single-hop clustering may generate a large number of cluster heads and eventually lead to the same problem as if there is no clustering.

In [49], Heinzelman et al. have proposed a distributed algorithm for wireless sensor networks (LEACH) in which the sensors randomly elect themselves as cluster heads with some probability and broadcast their decisions. The remaining sensors join the cluster of the cluster head that requires minimum communication energy. This algorithm allows only 1-hop clusters to be formed. LEACH assumes that all nodes are within communication range of each other and the base station (i.e. complete graph). LEACH clustering terminates in a constant number of iterations (like OK), but it does not guarantee good cluster head distribution and assumes uniform energy consumption for cluster heads [88].

Few papers have addressed the problem of multi-hop (k -hop) clustering [7, 9, 13]. These algorithms are mostly heuristic in nature and aim at generating the minimum number of disjoint clusters such that any node in any cluster is at most k hops away from the cluster head. For example, the algorithm described in [13] constructs clusters such that all the nodes within $R/2$ hops of a cluster head belong to that cluster head and the farthest distance of any node from its cluster head is $3.5R$ hops where R is an input parameter to the algorithm. With high probability, a network cover is constructed in $O(R)$ rounds; the communication cost is $O(R^3)$. The OK clustering algorithm has a much lower communication overhead. In [7], the authors presented the Max-Min heuristic to form non-overlapping k -clusters in a wireless ad hoc network. Nodes are assumed to have non-deterministic mobility pattern. Clusters are formed by broadcasting node identities along the wireless links. When the heuristic terminates, a node either becomes a cluster head, or is at most k wireless hops away from its cluster head. The value of k is a parameter of

the heuristic. Although the Max-Min algorithm generates k -hop clusters with a run-time of $O(k)$ rounds, it does not ensure that the energy used in communicating information to the information center is minimized. Both OK and MaxMin have $O(k)$ iterations. However, OK needs exactly $2k$ iterations to terminate but MaxMin needs at least $2k$ iterations. This means that the communication overhead is reduced in OK compared with MaxMin. In case of sensor networks, this directly affects the energy level of the node. In [9], the authors proposed a LEACH-like randomized clustering algorithm for organizing the sensors, in a wireless sensor network, in a hierarchy of clusters with an objective of minimizing the energy spent in communicating the information to the processing center (*base station*). They used results from stochastic geometry to obtain analytically the optimal number of cluster heads at each level of clustering.

None of the above algorithms construct overlapping clusters. Most of these algorithms are heuristic in nature and their aim is either to generate the minimum number of multi-hop clusters [7] or to minimize the energy spent in the network [9]. To the best of our knowledge, this is the first document to discuss the problem of overlapping multi-hop clustering. We show that constructing the minimum overlapping k -hop dominating set in an ad hoc network is NP-complete. Then we propose OK, a randomized multi-hop distributed algorithm to solve the problem. The nodes randomly elect themselves as cluster heads with some probability p . The clustering process terminates in $O(1)$ iterations, independent of the network diameter, and does not depend on the network topology or size. OK operates in quasi-stationary networks where nodes are location-unaware and have equal significance. The protocol incurs low overhead in terms of processing cycles and messages exchanged. We also analyze the effect of different parameters (e.g. cluster

radius, network connectivity, cluster head probability) on the performance of the clustering algorithm in terms of communication overhead, node coverage, and average cluster size.

OK is similar to the clustering algorithm described in [9] since both algorithms belong to the class of randomized multi-hop clustering. In [9], the main focus of the work was to find the optimal number of cluster heads at each level of clustering analytically, and apply this recursively to generate one or more levels of clustering. However, our main focus is to generate overlapping clusters with certain overlapping degree. Our main contributions are (i) to formalize the problem of overlapping multi-hop clustering; (ii) extend the work in [9] to meet the design goals; (iii) show how to tune the parameters (p and k) given to the algorithm in order to achieve the design goals; (iv) give analytical models to formulate the problem. In [9], the cluster radius (k) was calculated analytically to minimize the energy. In OK, the cluster radius is a parameter that can be tuned to increase overlapping degree between clusters, or to decrease the cluster size (load balancing), or to decrease communication overhead.

4.3 Problem Formulation

An ad-hoc network can be modelled as a graph $G = (V, E)$, where two nodes are connected by an edge if they can communicate with each other. If all nodes are located in the plane and have the same *transmission range* (T_r), then G is called a *unit disk graph*. We will start by describing the considered system model. Then, we will review a number of definitions from graph theory that will be used in the problem formulation. Finally,

we will formulate the overlapping k -hop clustering problem as an extension to the k -dominating set problem.

4.3.1 Definitions

Let n denote the number of vertices (nodes) and e denote the number of edges. That is, $n = |V|$ and $e = |E|$.

- *Open Neighbor Set*, $N(u) = \{v \mid (u, v) \in E\}$, is the set of vertices that are neighbors of u . For a set of nodes S , $N(S) = \bigcup_{u \in S} N(u)$.
- *Closed Neighbor Set*, $N[u] = N(u) \cup \{u\}$, is the set of neighbors of u and u itself. For a set of nodes S , $N[S] = \bigcup_{u \in S} N[u] = N(S) \cup S$.
- *Node Degree*, $\text{deg}(u) = |N(u)|$.
- *Graph Distance*, $d_G(u, v)$, the distance between two vertices u and v is the minimum number of edges in a $u - v$ path.
- *Graph Power*, the k th power of a graph G (G^k) is a graph with the same set of vertices as G and an edge between two vertices iff there is a path of length at most k between them [82]. Given $G = (V, E)$ then $G^k = (V, E^k)$ where $E^k = \{(u, v) \mid u, v \in V \text{ and } d_G(u, v) \leq k\}$.
- *Dominating Set*, S , is defined as a subset of V such that each vertex in $V - S$ is adjacent to at least one vertex in S . Thus, every MIS is a dominating set. However, since vertices in a dominating set may be adjacent to each other, not every

dominating set is an MIS. Finding a minimum-sized dominating set or MDS is NP-Hard [44].

- *Minimum Dominating Set (MDS)* is the dominating set with minimum cardinality. Each MIS is also an MDS. Finding the MDS is also NP-Hard [44].

The above definitions can be generalized for the multi-hop (k -hop) case as follows:

- *k -Connected Set, S* , a set S is said to be k connected if each vertex in S is within distance k from at least one other vertex in S , where $k \geq 1$ is an integer.
- *Open k -Neighbor Set, $N_k(u) = \{v | d_G(u, v) \leq k\}$* , is the set of vertices, different from u , that are at distance at most k from u . For a set of nodes S , $N_k(S) = \bigcup_{u \in S} N_k(u)$.
- *Closed k -Neighbor Set, $N_k[u] = N_k(u) \cup \{u\}$* , is the set of k -neighbors of u and u itself. If u is a cluster head, then $N_k[u]$ is the set of all vertices in the cluster and $|N_k[u]|$ is the cluster size. For a set of nodes S , $N_k[S] = \bigcup_{u \in S} N_k[u] = N_k(S) \cup S$.
- *Node k -Degree, $deg_k(u) = |N_k(u)|$* .
- *k -Dominating Set (KDS) OR Distance Domination, S* , is defined as a subset of V such that each vertex in $V - S$ is within distance k from at least one vertex in S , where $k \geq 1$ is an integer. That is $N_k[S] = V$.
- *The Overlapping Graph, G_S* , Let S be a KDS, then the overlapping graph is the weighted graph induced by S as follows:

1. The set of vertices are S .

2. An edge exists between two vertices u, v iff $N_k[u] \cap N_k[v] \geq \omega$, where ω is some threshold representing the minimal intersection.

The edge weights can be calculated according to different design goals. In section 6.3 we discuss different heuristics to calculate the weights. The overlapping graph could be undirected or directed graph based on how the weights are calculated. In the remainder of this chapter, we shall assume that the weights correspond to the overlapping degree between adjacent clusters; hence, the overlapping graph will be undirected graph.

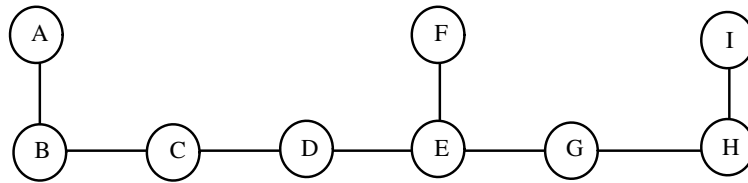


Figure 4.1: Illustrative Example

To clarify the above definitions, we will use the graph in Fig. 4.1 as an illustrative example. For the graph shown, $S_1 = \{A, G\}$ is a 2-dominating set (2DS) and it is also a 2-independent set (2IS); hence, S_1 is a 2-independent-dominating set (2IDS). The set $S_2 = \{C, D, E, G\}$ a 2-connected-dominating set (2CDS). The set is $S_3 = \{C, G\}$ is a 2-weakly-connected-dominating set (2WCDS). The set $S_4 = \{C, E, H\}$ a total 2-dominating set.

4.3.2 The Overlapped K-hop (OK) Clustering Problem

Given an ad-hoc network that is modelled as a unit disk graph $G = (V, E)$, the OK clustering problem can be formulated as finding the set of nodes S that satisfies the following conditions:

1. *Coverage Condition.* S is a KDS. This means that each node is either a cluster head or within k hops from a cluster head (i.e. $N_k[S] = V$).
2. *Overlapping Graph Connectivity Condition.* The overlapping graph induced by S , G_S , is connected. This condition implies that for each cluster head node u , there is at least one other cluster head node v such that the two clusters are overlapped with degree ≥ 1 (i.e. there is no isolated vertices in the overlapping graph). Moreover, there is at least one spanning tree that connects all the cluster head nodes together.

Finding the minimum KDS (*MKDS*) is a nice design goal to achieve. Minimizing the cardinality of the computed KDS can help to decrease the control overhead since broadcasting for node discovery and topology maintenance is restricted to a small subset of nodes. However, from a computational point of view, the problem of finding the minimum KDS (*MKDS*) is very difficult. In fact there is no known efficient centralized algorithm for solving this problem and a corresponding decision problem is NP-hard [48]. Even if the graph G belongs to certain special classes of graphs (for example if G is bipartite or chordal graph), the problem remains NP-hard [15]. The *MKDS* remains also NP-hard for unit-disk graphs as the case in wireless ad-hoc networks. Further aspects of the commutability of *MKDS* are discussed in [48, 24].

In [51], the authors described a centralized algorithm that finds a KDS of cardinality at most $n/(k+1)$. The algorithm firsts creates a rooted spanning tree from the original network topology. Then, an iterative labelling strategy is used to classify the nodes in the tree to be either dominator (cluster head) or dominated (non-cluster head). In [50], the authors described another centralized algorithm for finding the total KDS such that the cardinality is bounded by $2n/(2k+1)$. Since both algorithms are centralized, the communication overhead is high in case of large-scale networks like sensor networks. There is no known efficient distributed algorithm for finding the *MKDS* with some performance bound. For example, the MaxMin heuristic [7] finds a KDS, however, there is no reported performance bound on the cardinality of the resulting KDS. Similarly, in [9] the objective is to find a KDS that minimize energy consumption and maximize network lifetime.

A related problem that has been widely investigated in the context of wireless networks is the problem of finding the minimum connected dominating set (*MCDS*). The *MCDS* problem can be viewed as a special case of *MKDS* problem when $k = 1$. The *MCDS* is NP-hard for general graphs and for unit-disk graphs in particular [33]. Although there are many applications for CDS in wireless networks [14], the primary application of CDS is the construction of virtual backbone (spine) in wireless ad hoc networks. In the last decade, many CDS construction algorithms have been proposed in the context of MANETs and sensor networks. These algorithms are either centralized [18, 19, 28, 46] or distributed [5, 3, 4, 27, 29, 20, 85]. The centralized approaches seek a minimum connected dominating set (*MCDS*) as their major design goal. Thus performance bounds are their primary design parameter. However, centralized algorithms have high communication overhead and time complexity. On the other hand, distributed algorithms seek a

connected dominating set (not necessarily the minimum) that provides a good resource conservation property. Thus performance bound is not their primary consideration. Instead, time complexity (specially when nodes are mobile) and message complexity is taken into consideration. Distributed algorithms have a time complexity of $O(n)$ and a message complexity of $O(n \log n)$ [3, 29] or $O(n)$ [4, 20]. This quicker execution time comes at a cost of a larger CDS. A more detailed analysis of the performance of those algorithms is discussed at [14].

Any of the distributed heuristics for finding a CDS can be modified to find a KDS. In this case, we need to construct a k -closure (a graph power of order k) on the original connectivity network graph before running any of the heuristics. Recall from section 3.2 that the k th power of the graph yields a modified graph in which nodes A and B are 1-hop neighbors if they were at most k -hops away in the actual topology graph. When any of the distributed CDS heuristics are run on this modified graph, they form clusters where each node is at most k wireless hops away from its cluster head. Constructing the k th power of a graph is $O(kn^3)$, where n is the number of vertices in the graph [82]. Even if we used Strassen's algorithm for matrix multiplication [34], the best performance in terms of floating point operations is $O(kn^{2.807})$. For sensor networks, this is considered very expensive not in terms of communication overhead only but also the Strassen's algorithm is difficult to implement efficiently because of the data structures required to maintain the array partitions [34, 55]. Moreover, we are still generating non-overlapping clusters! Modifying an existing distributed CDS algorithm, to generate a KDS in a distributed randomized fashion, is a challenging problem in itself. We leave this as a future work.

The problem of overlapping clusters is totally new. There was no formulation of the problem in the literature. So there is no known algorithm that satisfies the three conditions described at the beginning of this section. The proposed OK clustering algorithm is a distributed simple randomized algorithm that meets the above three conditions with high probability. The main design goal behind the proposed algorithm is not to find the minimum KDS. Thus performance bound is not the primary consideration. Instead, we are more concerned about time complexity, processing complexity, and message complexity. We will show that by tuning some of the protocol parameters (k , p , node density), we can generate overlapping multi-hop clusters that satisfy the above three conditions with high probability. OK is scalable; the clustering formation terminates in a constant time $O(k)$ regardless of the network topology or size. The protocol incurs low overhead in terms of processing cycles and messages exchanged. OK assumes a quasi-stationary network where nodes are location-unaware and have equal significance. No synchronization is needed between nodes. In general, OK will produce a an overlapping KDS with the goal of minimizing the overall communication overhead, and processing complexity.

4.4 The OK Protocol Architecture

In this section we describe the operations of the OK protocol in more detail. The essential operation in any clustering protocol is to select a set of cluster heads among the nodes in the network, and cluster the rest of the nodes with these heads. OK does this in a distributed fashion, where nodes make autonomous decisions without any centralized control. The algorithm initially assumes that each sensor in the network becomes a cluster

head (*CH*) with probability p . Each cluster head then advertises itself as a cluster head to the sensors within its radio range. This advertisement is forwarded to all sensors that are no more than k hops away from the *CH*. Any sensor that receives such advertisements joins the cluster even if it already belongs to another cluster. Any sensor that is neither a *CH* nor has joined any cluster itself becomes a *CH*. Since the advertisement forwarding is limited to k hops, if a sensor does not receive a *CH* advertisement within time duration t_1 (where t_1 units is a value greater than the time required for data to reach the cluster head from any sensor k hops away), it can infer that it is not within k hops of any cluster head and hence become a *CH*. We assume that each cluster has a unique identifier, which is the node identifier of the cluster head. The flowchart of the OK algorithm is shown in Fig. 4.2. Each node maintains a table that stores information about the clusters known to this node. If the table contains more than one entry, this means that the node is a *boundary node*. Each cluster head maintains a list of all cluster members, a list of adjacent clusters, and a list of boundary nodes to reach those clusters. There can be multiple boundary nodes between overlapping clusters. Moreover, a node can be a boundary node for more than two overlapping clusters. In the remainder of this section, we first discuss the necessary data structures to be maintained at each node for the clustering protocol. We also discuss the message formats and the timers maintained by each node. We then explain the cluster formation protocol and give pseudo-code. Finally, we prove that the protocol meets its design goals.

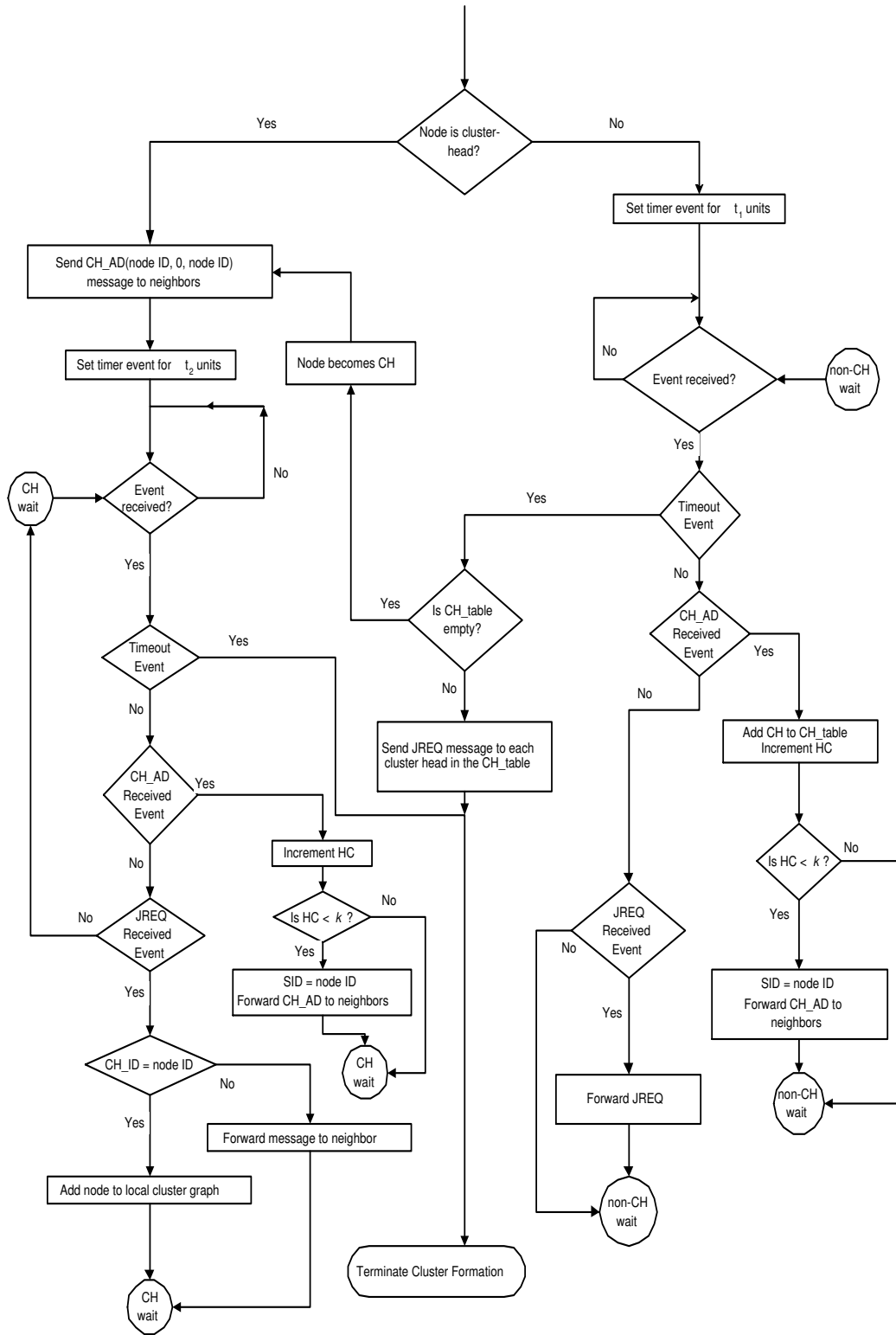


Figure 4.2: Flowchart of the OK cluster formation algorithm

4.4.1 Data Structures

Each node maintains the following variables:

- Node ID (*NID*): A unique ID assigned to each node before deploying the network.
- *Status*: {CH, NCH}. The status of the node. A node can be either a cluster head (CH) or a non-CH (NCH). Initially all nodes are set to NCH.
- Node Degree (*d*): The number of 1-hop neighbors. Calculated after discovering the number of neighbors.
- Reliable Ranges (*RR*): The number of reliable ranges known to the node. $RR \leq d$. Initialized during range estimation phase.
- Local Cluster Graph (*LCG*): $LCG = (V; E)$, a weighted undirected graph maintained by CH nodes corresponding to the local cluster that belongs to this CH node. The edge weight (w_{ij}) represents the range measurement between nodes i and j . Initially *LCG* consists of the CH node and all one-hop neighbors that it hears from during range estimation phase, i.e. $|V| = d+1$ and $|E| = d$, where d is the CH node degree. The LCG is very important for the localization purposes since it contains all range measurements between nodes. The LCG is used in the LLD phase to build the local map of the cluster and in the GLD phase to build the global network topology.
- Adjacent Clusters Table (*AC_table*): A table maintained by CH nodes to store information about adjacent clusters. The table consists of tuples of the form (*CHID*,

BN), where $CHID$ is the CH node ID, and BN is a list of *boundary node* IDs. Initially the table is empty.

- Cluster Heads Table (CH_table): A table maintained by each node to store information about the clusters known to this node. If the table contains more than one entry, this means that the node is a *boundary node*. The table consists of tuples of the form $(CHID, HC, prev)$, where $CHID$ is the CH node ID, HC is the number of hops leading to this cluster head, and $prev$ is the node ID of a 1-hop neighbor node that can lead to the CH node of this cluster using minimum number of hops. The table acts as a routing table where the $CHID$ field uniquely identifies a route to a CH node. Initially the table is empty.

4.4.2 Messages

There are two types of messages:

- CH advertisement (CH_AD) message: This is the message broadcast by a CH node to advertise its existence. It has the form $(CH_AD, SID, CHID, HC)$, where SID is the sender node ID, $CHID$ is cluster head ID, and HC is the number of hops leading to the CH node. The SID field is used to update the $CH_table.prev$ field such that each node knows a unique path to the cluster head. The HC field is used to limit the flooding of the CH_AD message to k hops. The CH_AD message has a fixed size.
- Join request ($JREQ$) message: This is a message sent by a node when it knows about the existence of a CH node and decides to join this cluster. To limit the flooding, the message is unicasted using the field $CH_table.prev$. The message

contains information about the range measurements to neighbors along with the clusters that this node can hear from. The message has the form (JREQ, RID , SID , $CHID$, d , $(NID, R_{SID,NID})_{1..nd}$, nc , $(CHID)_{0..nc}$), where RID is the receiver ID¹, SID is the ID of the node that will join the cluster, $CHID$ is the ID of the CH node responsible for this cluster, d is the node degree, $(NID, R_{SID,NID})_{1..nd}$ are one or more couples containing information about the range measurements between this node and its 1-hop neighbors, nc is the number of clusters that this node can hear from them ($=|AC_table|$), and $(CHID)_{0..nc}$ are 0 or more clusters that this node can hear from. Notice that the size of the JREQ message is variable and depends on the number of clusters (nc) and the node degree (d).

4.4.3 Timers

Each node maintains the following timers²:

- CH_AD_WAIT timer. This timer is set by a non-CH node. It represents the maximum time that a node should wait for CH advertisement messages. It is equal to $t_1 = t(k) + \delta$, where $t(k)$ is the time needed for a message to travel k hops and δ is the maximum time needed for any node to finish bootstrapping and start executing the OK protocol. In our simulator, we assume that all the CH nodes will finish bootstrapping and start transmitting CH_AD messages within $t(k)/2$ time units. Hence, we set δ to be $t(k)/2$.

¹This equals to $CH_table.prev$ corresponding to the $CH_table.CHID$.

²We assume a timer that is set to a certain number of units and fires once.

- JREQ_WAIT timer. This timer is set by a non-CH node. It represents the maximum time that a node should wait for JREQ messages to forward to CH nodes. It is also equal to t_1 .
- CH_WAIT_SHORT timer. This timer is set by a CH node that was initially an NCH node but change status. It represents the time a CH node should wait for JREQ messages before terminating the cluster formation phase. It is approximately equal to $t_2 = 2 * t(k) + \delta$.
- CH_WAIT_LONG timer. This timer is set by a CH node. It represents the maximum time that a CH node should wait for JREQ messages before terminating the clustering algorithm. It is longer than the CH_WAIT_SHORT timer and is approximately equal to $t_3 = 3 * t(k) + \delta$.

The events of the OK clustering algorithm are listed in table 4.1. A finite state machine for the protocol is given in Fig. 4.3. The activities of the OK clustering algorithm are shown in Fig. 4.4, using an event-based notation.

4.5 Performance Evaluation

We have validated the OK clustering algorithm using simulation. The OK clustering algorithm was implemented using MATLAB 6.1 release 12.1. Initially, each node is assigned a unique node id. There are four parameters used in our simulation:

1. *Network size (n)*: the number of sensor nodes in the network. Since all the simulation experiments assume a square area of side length l , changing the network

Event Name	Description
Initialization()	An event executed once to initialize the status of the node.
CH_AD_Received (<i>SID</i> , <i>CHID</i> , <i>HC</i>)	An event triggered when CH_AD message is received.
JREQ_Received (<i>RID</i> , <i>SID</i> , <i>CHID</i> , <i>nd</i> , (<i>NID</i> , <i>RSID</i> , <i>NID</i>) _{1..nd} , <i>nc</i> , (<i>CHID</i>) _{0..nc})	An event triggered when JREQ message is received.
ChangeStatus	An event triggered when the CH_AD_WAIT timer fires indicating that an NCH node should either change its status to CH node or join a cluster if any.
EndClusterFormationPhase	An event triggered when the JREQ_WAIT timer fires indicating that a CH node should terminate the clustering phase and start the Local Location Discovery (LLD) phase.

Table 4.1: Events summary of the OK clustering algorithm

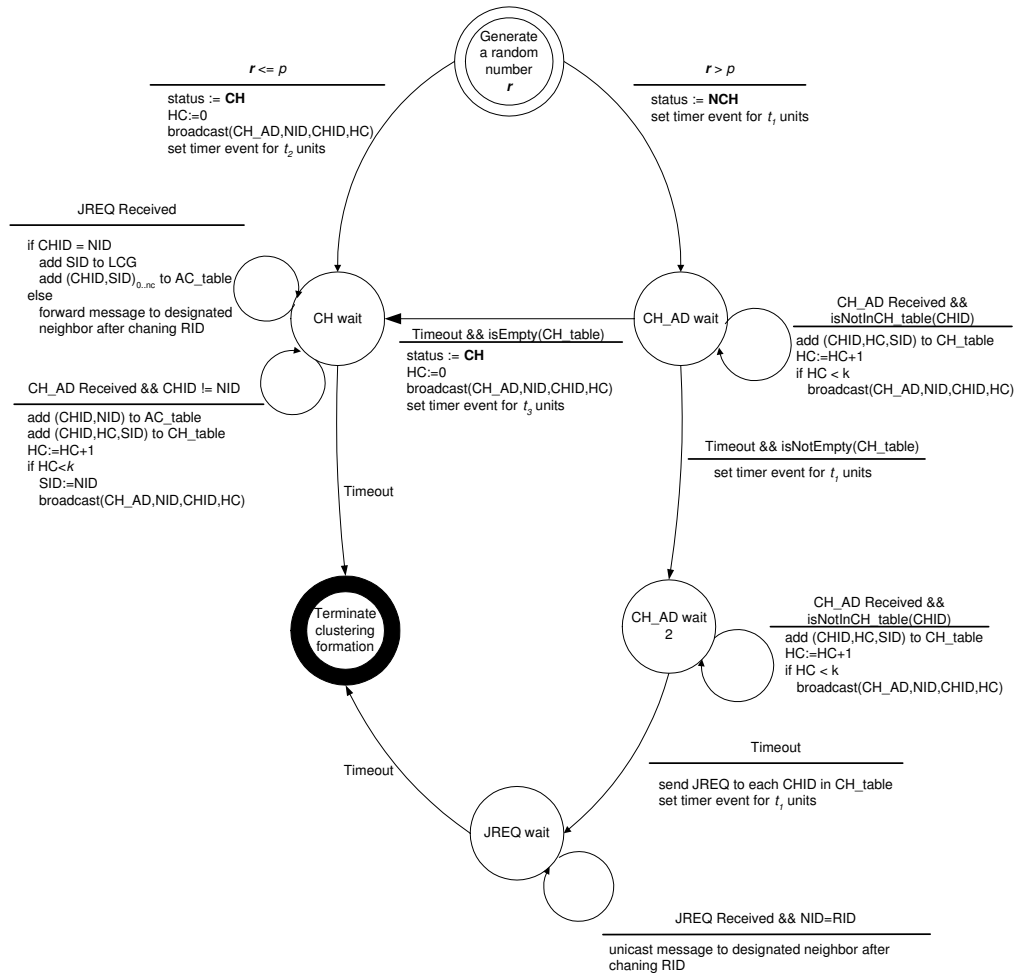


Figure 4.3: Finite state machine of the OK protocol

```

Initialization() // executed once
1. ac:
2. r = generate random number from 0..1;
3. if r < p then
4.   status := CH;
5.   broadcast (CH_AD, NID, NID, 1);
6.   set CH_WAIT timer;
7. else
8.   status := NCH;
9.   set CH_AD_WAIT timer;

CH_AD_Received (SID, CHID, HC)
10. ac: if status = NCH
11.   if CHID is not in the CH_table
12.     Add (CHID, HC, SID) to CH_table;
13.     if HC < k
14.       HC := HC + 1;
15.       broadcast (CH_AD, NID, CHID, HC);
16.     // else HC ≥ k, do not forward the message more than k hops
17.     // else you have already heard of this cluster, do nothing
18. else
19.   // node is a CH node
20.   if CHID = NID
21.     discard the message; // This is an echo message
22.   if CHID is not in the AC_table
23.     Add (CHID, NID) to AC_table;
24.     Add (CHID, HC, SID) to CH_table;
25.     if HC < k
26.       HC := HC + 1;
27.       broadcast (CH_AD, NID, CHID, HC);
28.     // else HC ≥ k, do not forward the message more than k hops
29.     // else you have already heard of this cluster, do nothing

JREQ_Received (RID, SID, CHID, nd, (NID, RSID, NID)1..nd, nc, (CHID)0..nc)
30. ac: if status = NCH
31.   if RID = NID
32.     RID := CH_table[CHID].prev;
33.     broadcast (JREQ, RID, SID, CHID, nd, (NID, RSID, NID)1..nd, nc, (CHID, cost)0..nc);
34.   // else do nothing to limit the flooding of JREQ message
35. else
36.   // node is a CH node
37.   if CHID = NID
38.     Add SID to the set of vertices in LCG;
39.     Add (NID, RSID, NID)1..nd to the set of edges in LCG;
40.     Add (CHID, cost, SID)0..nc to the AC_table;
41.   else
42.     RID := CH_table[CHID].prev;
43.     broadcast (JREQ, RID, SID, CHID, nd, (NID, RSID, NID)1..nd, nc, (CHID, cost)0..nc);

EndClusterFormationPhase
44. ec: (CH_WAIT timer fires && status = CH)
      OR (CH_WAIT_SHORT timer fires && status = CH)
      OR (JREQ_WAIT timer fires && status = NCH)
45. ac: Start the Local Location Discovery (LLD) phase using information stored in LCG and AC_table.

ChangeStatus
46. ec: CH_AD_WAIT timer fires. // for NCH node
47. ac: if CH_table empty
48.   status := CH;
49.   broadcast (CH_AD, NID, NID, 1);
50.   set CH_WAIT_SHORT timer;
51. else
52.   for all CHID in CH_table
53.     RID := CH_table[CHID].prev;
54.     broadcast (JREQ, RID, NID, CHID, (NID, RSID, NID)1..d, (CHID)0..m);
55.   set JREQ_WAIT timer;

```

Figure 4.4: The OK Algorithm

size will implicitly change the node density in the network (μ). Node density (μ) is defined to be the number of nodes in unit area:

$$\mu = n/l^2 \quad (4.1)$$

2. *Cluster radius (k):* the maximum graph distance between any node in the cluster and the cluster head. Recall from section 3.2 that the graph distance between two vertices u and v , $d_G(u, v)$, is the minimum number of edges in a $u - v$ path. Hence, if u is a CH node, then $k = \max_{v \in N_k(u)} (dist_G(u, v))$.

3. *Average Node Degree (d):* the average node degree in the network. Recall from section 4.3.1 that the node degree of a node u , is the number of nodes that are neighbors of u . Node degree is a function of the node transmission range (T_r). Assuming that n sensor nodes are uniformly distributed over a square field of side l , the probability $P(d)$ of a node u having degree d is given by binomial distribution [75]:

$$P(d) = P_r^d (1 - P_r)^{n-d-1} \binom{n-1}{d} \quad (4.2)$$

where P_r is the probability of being within the transmission range T_r from node u

$$P_r = \frac{\pi \cdot T_r^2}{l^2} \quad (4.3)$$

For large values of n tending to infinity, the above binomial distribution converges to a Poisson distribution:

$$P(d) = \frac{\lambda^d}{d!} e^{-\lambda} \quad (4.4)$$

where $\lambda = nP_r$ is the average node degree. Hence, the relation between the average node degree (d) and the transmission range (T_r) of a node is given by:

$$d = nP_r = \frac{n \cdot \pi \cdot T_r^2}{l^2} = \mu \cdot \pi \cdot T_r^2 \quad (4.5)$$

We will use the above equation frequently to map between average node degree and transmission range.

4. The cluster head probability (p). Since each node decides randomly to be a cluster head with probability p , then the average number of clusters is pn . Hence, increasing p will increase the number of clusters in the network.

All experiments were performed over 150 different topologies representing different network sizes (n) ranging from 50 to 800 sensor nodes. The nodes were randomly placed according to a uniform distribution on a 100x100 area. For each topology, the transmission range of each node (T_r) was varied in order to achieve different average *node degree* (d) ranging from 7 to 21. In a wireless ad-hoc network with a uniform distribution of nodes, in order to guarantee global network connectivity, the average node degree should be at least 6 [61]. Hence, we chose the minimum average node degree to be 7. The cluster radius (k) ranges from 1 to 5. The cluster head probability (p) was varied from 0.05 to 0.5. For each topology, since cluster heads are chosen randomly, we repeat the experiment 30 times, each time with a different random set of cluster heads. To evaluate the performance of the OK clustering algorithm, we use the following performance metrics:

1. *Percentage of Covered Nodes (CN)*: this metric tests if the generated clusters satisfy the *coverage condition* as defined in section 4.3.2. *CN* is defined as the percentage

of nodes that are either cluster heads or within k -hops from a cluster head after the first wave of CH advertisement is propagated through the network (i.e. after $t(k)$ time units where $t(k)$ is the time needed for a message to be forwarded for k hops). We will prove in section 4.7 (*lemma 4.2*) that after $3t(k) + \delta$, the OK clustering terminates and each node is either CH or NCH.

2. *The Average Overlapping Degree (AOD)*: this metric tests if the generated clusters satisfy the *overlapping condition* as defined in section 4.3.2. AOD is defined as the average overlapping degree between any two overlapping clusters in the network. Assume that u, v are any two cluster head (CH) nodes. Then the overlapping degree between the two corresponding clusters (\mathbf{O}) is a discrete random variable where $\mathbf{O} = |N_k[u] \cap N_k[v]|$ and $N_k[u] \cap N_k[v] \neq \emptyset$. Notice that the overlapping degree is defined only for overlapping clusters (i.e. the random variable \mathbf{O} can not take the value 0). We define *AOD* as the mean of this random variable \mathbf{O} (i.e. $AOD = E(\mathbf{O})$).
3. *The Connectivity Ratio (CR)*: this metric tests if the generated clusters satisfy the *connectivity condition* as defined in section 4.3.2. Let S be the set of CH nodes. Let G_S be the undirected graph induced by S such that an edge exists between two nodes $u, v \in S$ if $dist_G(u, v) < 2k$ (i.e the two corresponding clusters overlap). Notice that G_S is not necessary a connected graph. Then the connectivity ratio (*CR*) is defined as ratio between the number of nodes in the largest spanning tree of G_S to the number of CH nodes ($|S|$). If $CR = 1$, this means that G_S is a connected graph.
4. *The Average Cluster Size (N_c)*: the average number of nodes per cluster taken over-

all clusters. If u is a CH node, then $N_c = |N_k[u]|$. We use this metric to show that OK generates equal-sized clusters, which is a desirable property to balance the load of control overhead between cluster head nodes.

5. *The Average Number of Edges per Cluster (E_c):* the average number of edges per cluster taken over all clusters. This metric is important for localization applications [89] since the number of edges in the graph affect the accuracy of the estimated node positions.
6. *The Average CLIQUE Factor per Cluster (CF):* the CLIQUE factor of a cluster measures how close the subgraph induced by cluster to a complete graph. The CF is calculated as follows:

$$CF = \frac{2 * E_c}{N_c * (N_c - 1)} \quad (4.6)$$

7. *Communication Overhead:* this metric measures the total energy spent in communication. Without loss of generality, it is assumed that the cost of transmitting 1 unit of data (byte) is 1 unit of energy. This is a valid assumption since we assume that all the nodes have a fixed transmission range.

The first three performance metrics measure how close is OK to meet the conditions listed in section 4.3.2. N_c , E_c , and CF give more insight into the size of each cluster. Finally, measuring the communication overhead shows how scalable the proposed algorithm is in terms of messages exchanged between nodes. For simplicity, we assume that the communication environment is contention-free and error-free; hence, sensors do not have to retransmit any data. The Multiple Access with Collision Avoidance (MACA) protocol [72] may be used to allow asynchronous communication while avoiding collisions.

MACA utilizes a Request To Send/Clear To Send (RTS/CTS) handshaking to avoid collision between nodes. Other MAC protocols such as TDMA [32] may be used to provide collision-free MAC layer communication.

Our main goals behind the simulation experiments are: (1) to show that with the careful selection of input parameters (p , k , d), the proposed clustering algorithm meets the conditions listed in section 4.3.2 with high probability; (2) to show that although we have overlapped clusters, the OK clustering still produces approximately equal-sized clusters; (3) to show that OK is scalable in terms of communication overhead. Since each of the above protocol parameters has a different effect on one of the performance metrics, we wanted to give a sensor network engineer a set of parameters to tune to achieve different design goals (minimize power consumption by playing with node transmission range, increase overlapping degree, reduce cluster size, increase inter-cluster connectivity, reduce number of clusters, reduce cluster formation time). In order to qualify the impact of the various parameters, we will try answering the following questions:

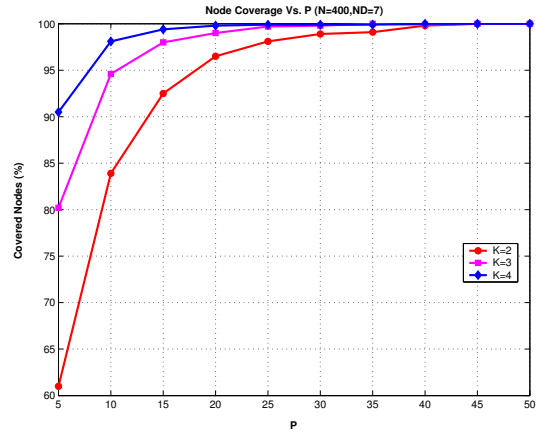
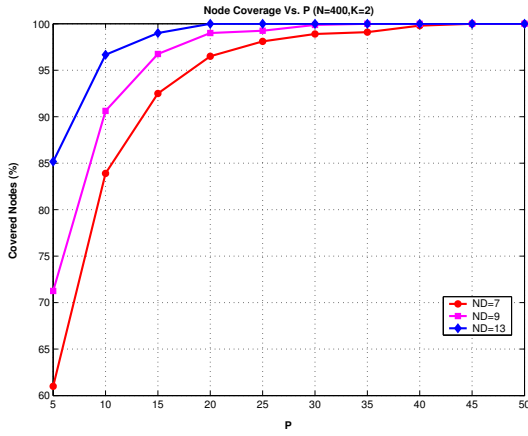
- Q1: What is the effect of different simulation parameters (k , d , p) on the percentage of covered nodes (CN) (section 4.5.1)?
- Q2: What is the effect of different simulation parameters (k , d , p) on the average overlapping degree (section 4.5.1)?
- Q3: What is the effect of different simulation parameters (k , d , p) on the connectivity ratio (section 4.5.1)?
- Q4: What is the effect of different simulation parameters (k , d , p) on N_c , E_c and CF (section 4.5.2)?

- Q5: What is the total energy spent in communication until the clustering protocol terminates (section 4.5.3)?
- Q6: Is the OK protocol scalable (section 4.5.3)?
- Q7: Given (n, k, d) , what are the best protocol parameters that guarantee that all the conditions discussed in section 4.3.2 are satisfied with high probability?

4.5.1 Coverage, Cluster Overlapping and Connectivity Ratio

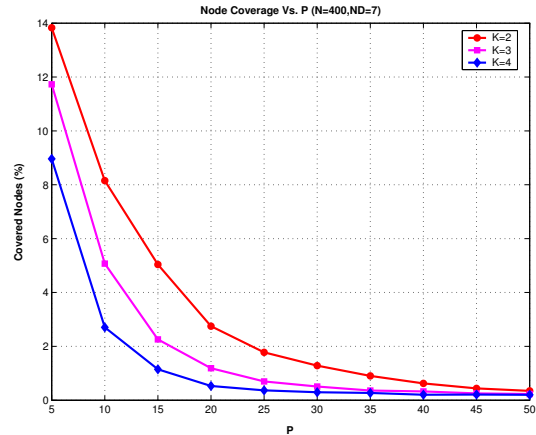
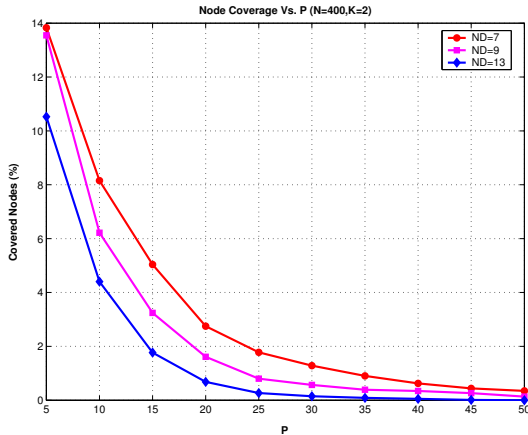
We start by studying the effect of cluster head probability (p) on the percentage of covered nodes (CN). From Fig. 4.5, we can see that increasing p increases the coverage almost exponentially specially for lower values of d (i.e. low transmission range). The standard deviation curves (Fig. 4.5(c) and 4.5(d)) show that the coverage is guaranteed within 2% for $p \geq 0.25$. It is also clear that for each combination of (k, d) , there is a minimum value for p that guarantees 100% coverage with high probability. We will discuss this in more details in section 5.4.

The impact of average node degree (d) on the percentage of covered nodes is shown in Fig. 4.6(a). Increasing d increases the coverage almost exponentially for lower values of k . For $k > 1$, increasing d above a certain threshold has almost no effect on the coverage. The standard deviation curve (Fig. 4.6(c)) shows that this is guaranteed within 1% with high probability for $d \geq 16$ and $k > 1$. In Fig. 4.6(b), the relation between cluster radius (k) and percentage of covered nodes is shown. Increasing k seems to increase the coverage exponentially. Again we can see from the standard deviation curve in Fig. 4.6(d) that the results are within 1% if $k > 2$ and $d \geq 9$. These values for k and d are very com-



(a) The impact of cluster head prob. (p) on the percentage of covered nodes (different d)

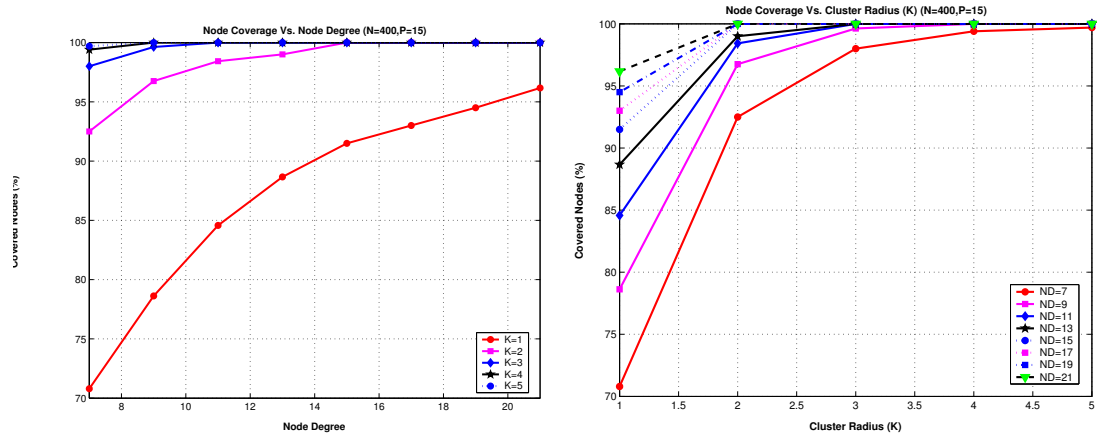
(b) The impact of cluster head prob. (p) on the percentage of covered nodes (different k)



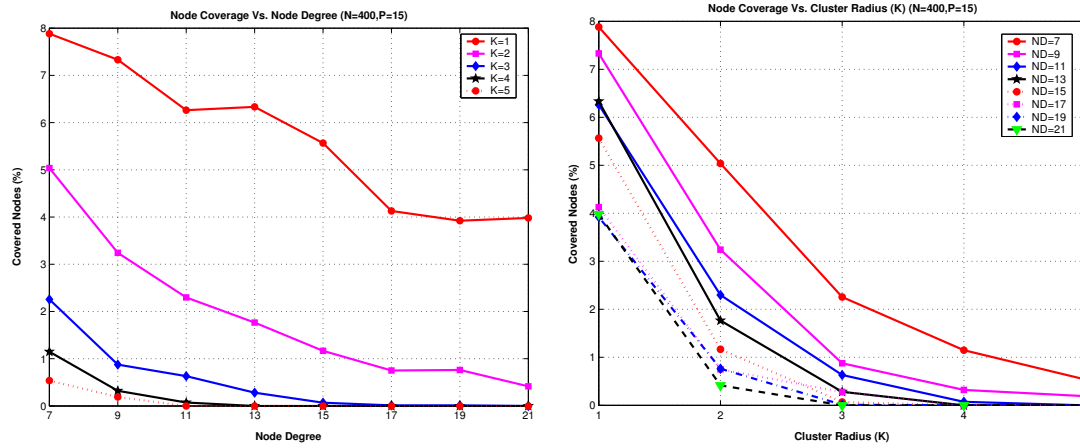
(c) The standard deviation of percentage of covered nodes as (p) increases (different d)

(d) The standard deviation of percentage of covered nodes as (p) increases (different k)

Figure 4.5: The relation between cluster head prob. (p) and percentage of covered nodes



(a) The impact of average node degree (d) on the (b) The effect of cluster radius (k) on the percentage of covered nodes



(c) The standard deviation of percentage of covered nodes as d increases (d) The standard deviation of percentage of covered nodes as k increases

Figure 4.6: The impact of average node degree (d) and cluster radius (k) on percentage of covered nodes

mon and realistic in sensor networks applications. As a summary, the effect of increasing d , k is the same. However, d is directly proportional to transmission range; hence it affects node energy dramatically. On the other hand, k is application dependent. For example, in routing protocols, increasing k will increase cluster size, and latency; in localization applications, increasing k will reduce the accuracy of the estimated node position. We will see later in section 4.5.3, that both k and d increase communication overhead. However, the communication overhead is proportional to k^3 as we will discuss in section 4.5.3. Finally, from the figures we can see that by careful selection of the parameters (p , d , k) we can guarantee 100% coverage with high probability. This means that each node is either a cluster head or belongs to at least one cluster (i.e. the *coverage condition* discussed in section 4.3.2 is satisfied with high probability).

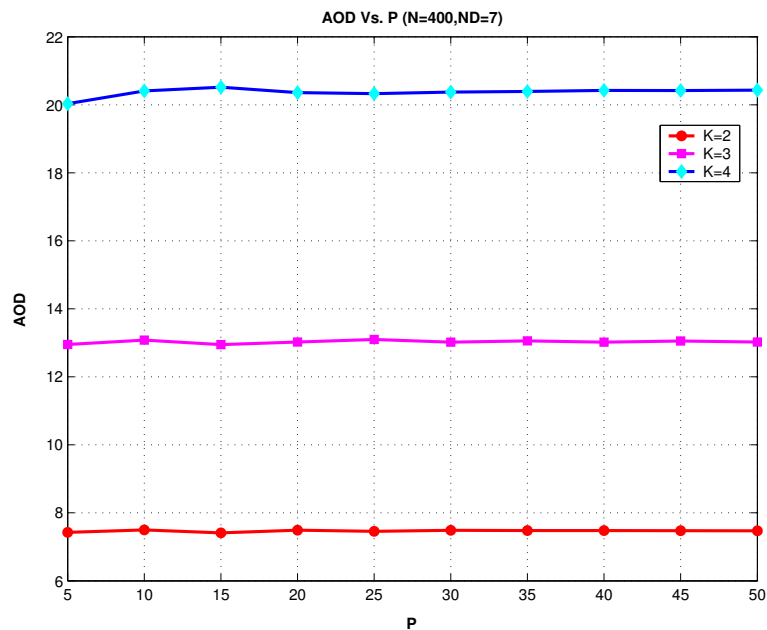
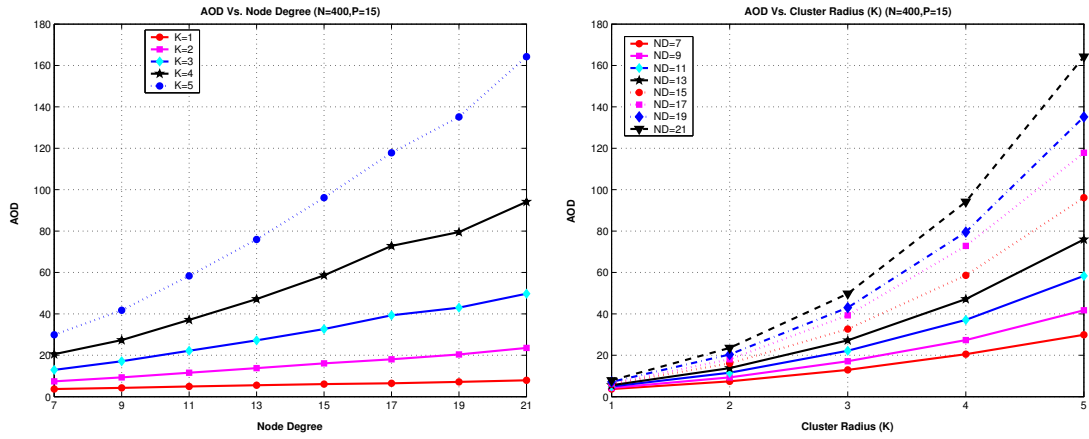
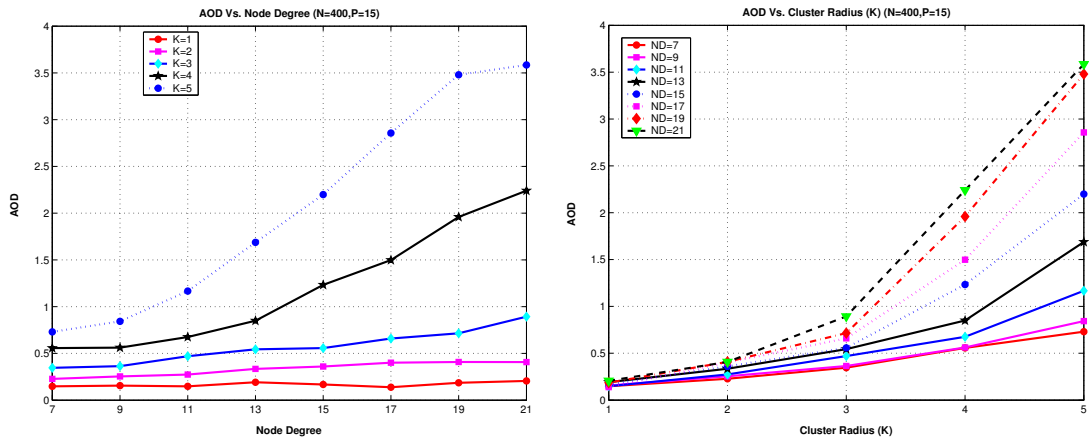


Figure 4.7: The cluster head prob. (p) has no effect on the average overlapping degree (AOD)



(a) The impact of average node degree (d) on the average overlapping degree (AOD) (b) The effect of cluster radius (k) on the average overlapping degree (AOD)

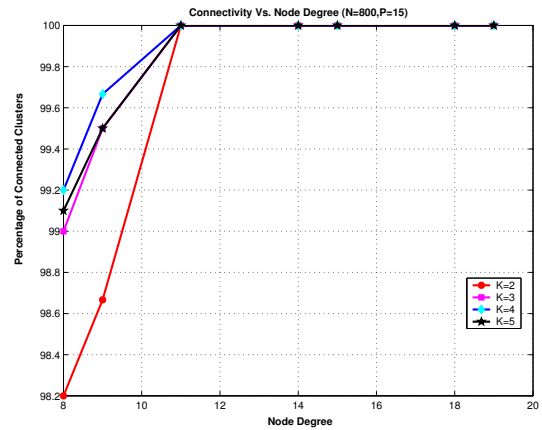
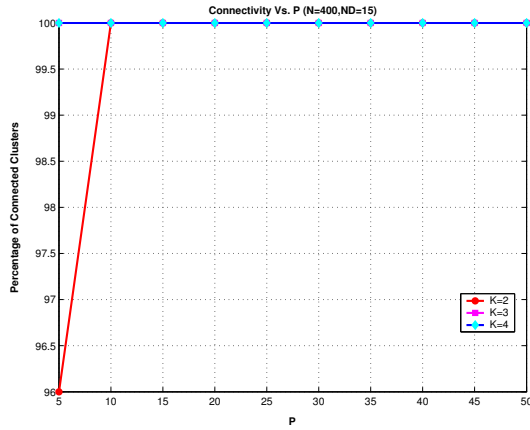


(c) The standard deviation of average overlapping degree (AOD) as d increases (d) The standard deviation of average overlapping degree (AOD) as k increases

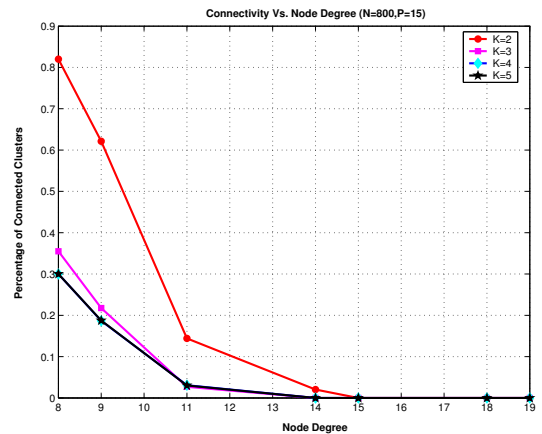
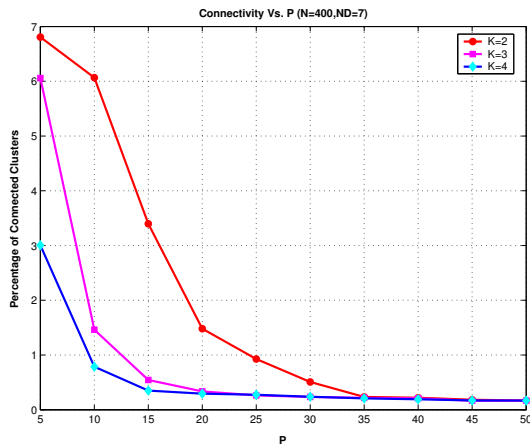
Figure 4.8: The impact of average node degree (d) and cluster radius (k) on average overlapping degree

We will now turn our discussion to the study of the average overlapping degree between clusters. Fig. 4.7 shows an interesting anomaly. Although one may think that increasing p (i.e. increasing number of cluster heads and hence clusters) should increase the average overlapping degree (AOD), the results showed that p has no effect on AOD regardless of the values of other parameters (d , k) and network size (n). We will prove analytically in section 4.6.3 that the AOD does not depend on p . This will leave us with only two parameters to play with to control the overlapping between clusters d , and k . As shown in Fig. 4.8(a), the AOD is linearly proportional with d . Notice that AOD can never exceed the network size n so the curve saturates at n . On the other hand, increasing the cluster radius (k) will increase the AOD quadratically as shown in Fig. 4.8(b). We will discuss analytically in more details the relation between AOD and d and k in section 4.6.3. Notice that for many applications, the required AOD between clusters should be below 10. For example in SALAM, an AOD of 3 is enough and in routing protocols having 10 gateway nodes between clusters is more than enough. It is clear that we can guarantee an AOD of more than 10 with high probability using small d (i.e. low transmission range) and small cluster radius ($k = 2$). This is confirmed also by the standard deviation curves, Fig. 4.8(c) and 4.8(d). We can clearly see from the curves that an AOD of at least 10 can be guaranteed with high probability if $k \geq 2$ and any $d > 6$.

Finally, to show that the OK protocol satisfies the *connectivity condition*, as defined in section 4.3.2, we study the connectivity between clusters. Fig. 4.9(a) shows the relation between connectivity ratio and p for different values of k . The figures show that with 15% of the nodes are cluster heads; we can have 100% connectivity with high probability. This means that for any cluster head, there is a path of less than $2k$ hops to at least another



(a) The relation between connectivity ratio (CR) and the cluster head prob. (p) (b) The relation between connectivity ratio (CR) and average node degree (d)



(c) The standard deviation of percentage of connected clusters as p increases (d) The standard deviation of percentage of connected clusters as d increases

Figure 4.9: The effect of the cluster head prob. (p) and average node degree (d) on percentage of connected clusters

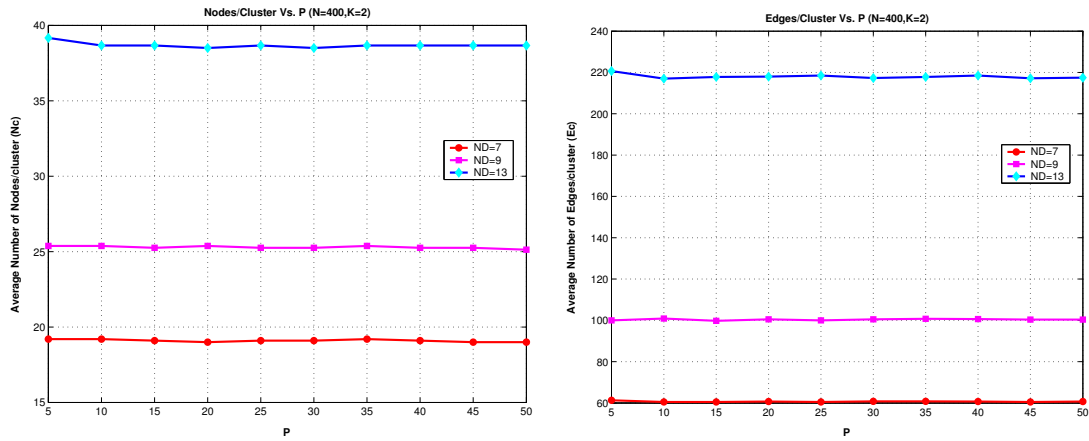
cluster head (i.e. there is at least one border node between the two clusters). We can see that this still holds for any value of k and $d > 10$ as shown in Fig. 4.9(b). The standard deviation curves (Fig. 4.9(c) and 4.9(d)) confirm the above results with high probability.

As a general conclusion, it is clear that the OK protocol satisfies with high probability the three conditions, defined in section 4.3.2. The cluster head probability (p) plays an important role in terms of coverage and connectivity between cluster. The average node degree (d) and the cluster radius (k) can be tuned to achieve a reasonable average overlapping degree between clusters regardless of p .

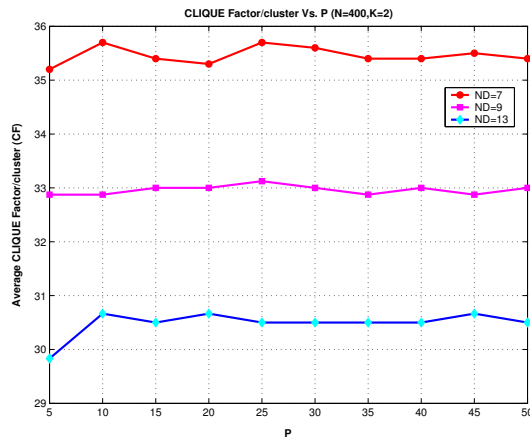
4.5.2 Cluster Size

In this section we will study the properties of the generated clusters in terms of average cluster size (N_c), average number of edges per cluster (E_c) and average CLIQUE factor (CF). Since the clusters are overlapping, increasing the number of clusters will not affect the cluster size. Hence, p has no effect on N_c , E_c and CF as shown in Fig. 4.10. On the other hand, increasing d increases N_c linearly, as shown in Fig. 4.11(a), and increases E_c quadratically (Fig. 4.11(b)). Substituting in Eq. 4.6, we can see why the CF is almost constant as d increases (Fig. 4.11(c)). A detailed analytical model for the average cluster size is discussed in section 4.6.2.

As a measure of load balancing, the standard deviation of average number of nodes per cluster is shown in Fig. 4.11(d). The figure shows very low standard deviation regardless of the values of d and k . This means that the OK protocol produces equal-sized clusters. The same facts can be concluded from the standard deviation curves of number



(a) The cluster head prob. (p) has no effect on the average number of nodes/cluster
 (b) The cluster head prob. (p) has no effect on the average number of edges/cluster



(c) The cluster head prob. (p) has no effect on the average CLIQUE factor (CF)

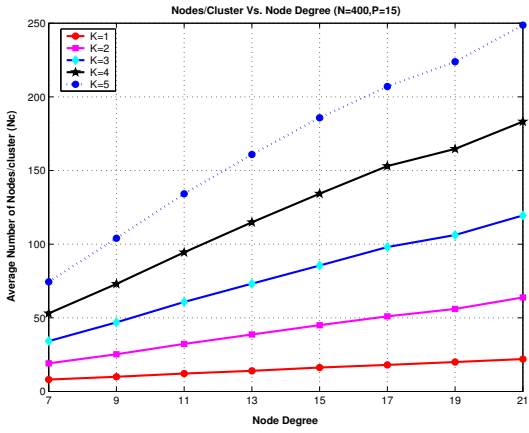
Figure 4.10: The cluster head prob. (p) has no effect on cluster size properties

of edges per cluster (Fig. 4.11(e)) and the average CLIQUE factor (Fig. 4.11(f)). From Fig. 4.12(a) and 4.12(b), we can see that both N_c and E_c are proportional with the square of the cluster radius (k^2). Hence, from Eq. 4.6, we can see why the average CLIQUE factor (CF) decreases quadratically as k increases (Fig. 4.12(c)). Again the standard deviation curves, Fig. 8 4.12(d), 4.12(e), 4.12(f), confirm that OK produces equal-sized clusters regardless of the values of d and k .

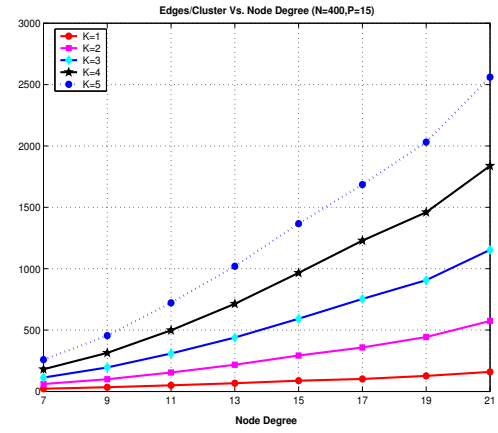
As a final conclusion, although the OK protocol generates overlapping clusters, the simulation results show that those clusters are equal-sized. Equal-sized clusters is a desirable property because it enables an even distribution of control (e.g., data processing, aggregation, storage load) over cluster heads; no cluster head is overburdened or underutilized. Moreover, the results show that the average cluster size can be controlled by tuning the average node degree (d) or the cluster radius k . A closed form for the upper bound of the average cluster size (N_c) as a function of d and k is given in section 4.6.2. Finally, the average number of edges and the intra-cluster connectivity, measured by the CF metric, can also be controlled by changing d and k . This is a desirable feature in SALAM as we will discuss in section 5.10.3.

4.5.3 Scalability

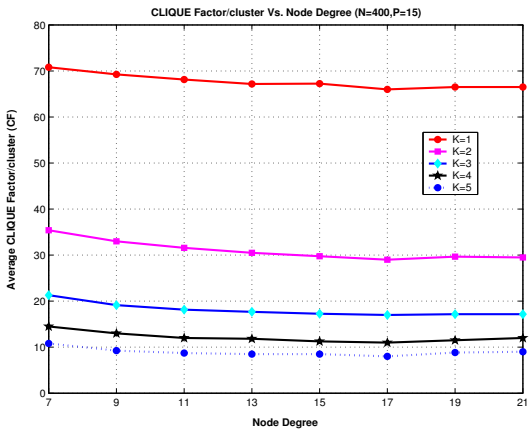
In this section we analyze the communication overhead of the OK clustering protocol and show that OK is scalable and energy efficient in terms of communication overhead. The total energy spent in communication is measured in terms of the number of bytes transmitted per node. Without loss of generality, it is assumed that the cost of transmitting



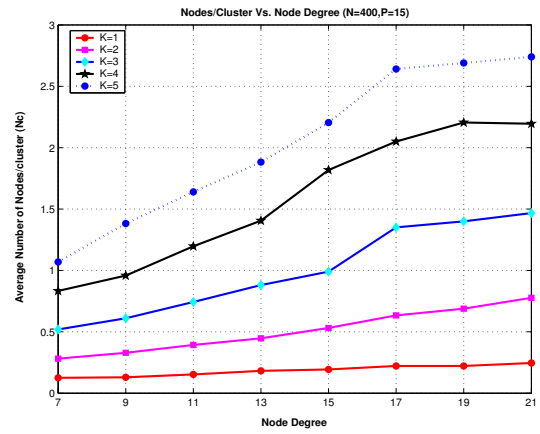
(a) The effect of average node degree (d) on number of nodes/cluster (N_c)



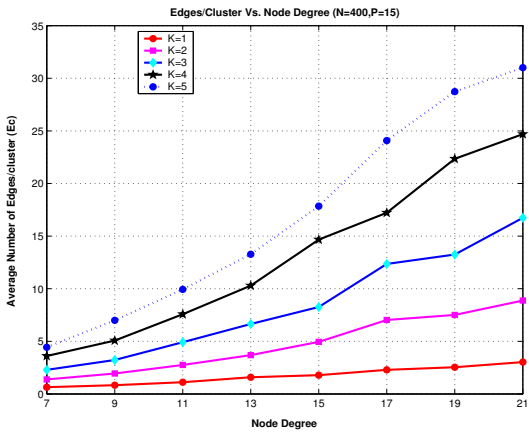
(b) The effect of average node degree (d) on number of edges/cluster (E_c)



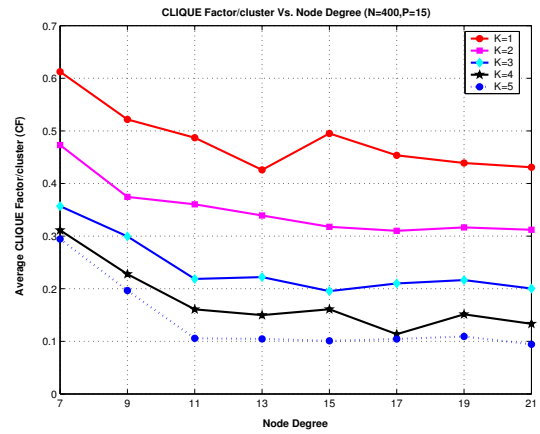
(c) The effect of average node degree (d) on the CLIQUE factor (CF)



(d) The standard deviation of the average number of nodes/cluster (N_c) as d increases

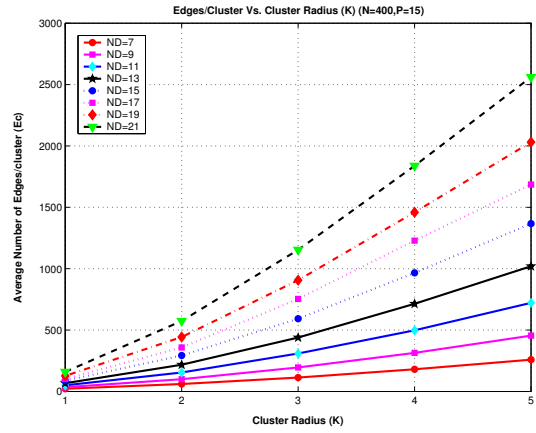
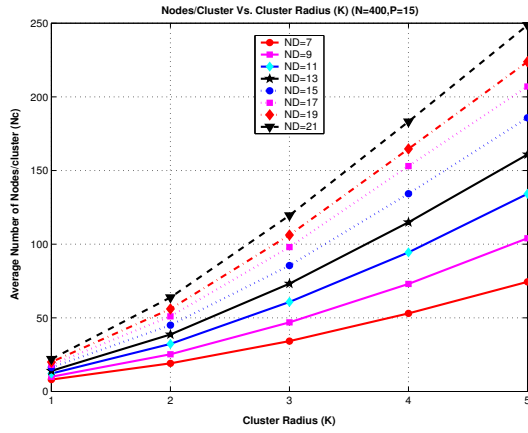


(e) The standard deviation of the average number of edges/cluster (E_c) as d increases



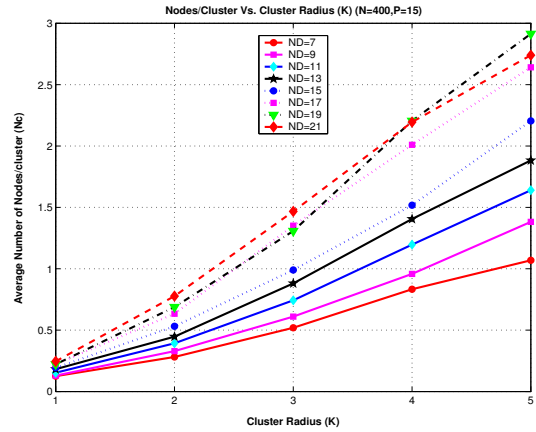
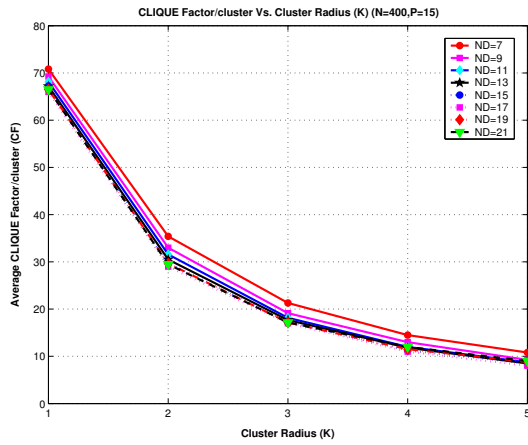
(f) The standard deviation of the average CLIQUE factor (CF) as d increases

Figure 4.11: The effect of average node degree on cluster size properties



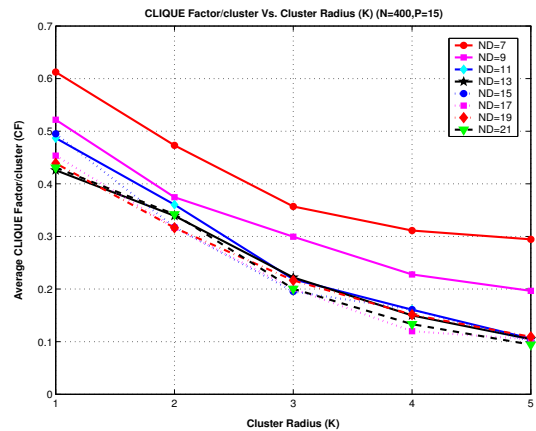
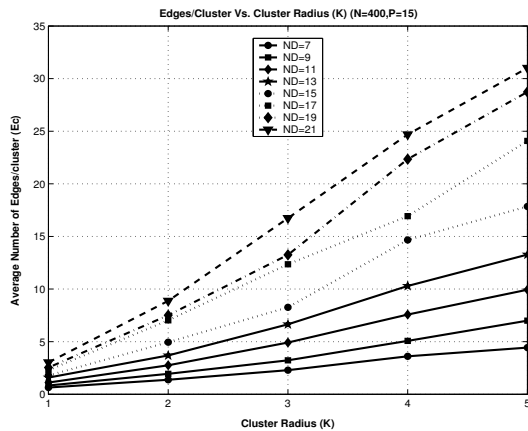
(a) The effect of cluster radius (k) on number of nodes/cluster (N_c)

(b) The effect of cluster radius (k) on number of edges/cluster (E_c)



(c) The effect of cluster radius (k) on the CLIQUE factor (CF)

(d) The standard deviation of the average number of nodes/cluster (N_c) as k increases



(e) The standard deviation of the average number of edges/cluster (E_c) as k increases

(f) The standard deviation of the average CLIQUE factor (CF) as k increases

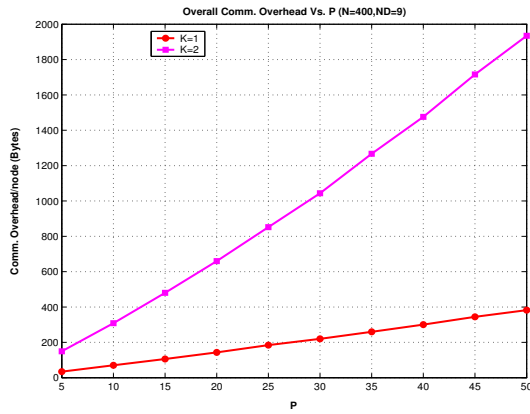
Figure 4.12: The effect of cluster radius on cluster size properties

1 unit of data (byte) is 1 unit of energy. This is a valid assumption since we assume that all the nodes have a fixed transmission range. We will start by describing the model used for estimating the communication overhead. Then we show the impact of different simulation parameters on the overall communication overhead and study the scalability of the OK protocol. An analytical model for the communication overhead is discussed in the next section (4.6.4).

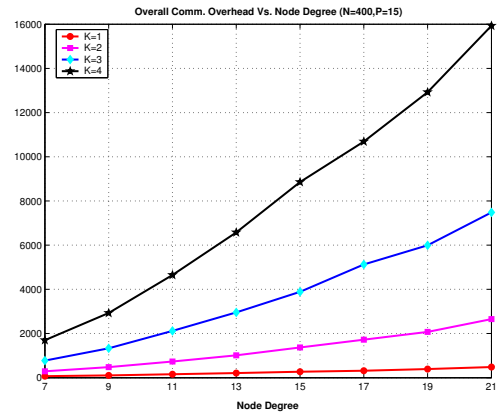
There are two phases in the OK protocol: the cluster head advertisement phase (CHAD phase) and the join request phase (JREQ phase). For each network topology i with network size n , we calculate the total number of bytes sent by all the nodes during the two phases ($TotalMsgSize(i)$). We then repeat the experiment over 900 different topologies, with the same network size n . Hence, the average number of bytes sent by all nodes ($avgTotalMsgSize$) is the mean of the vector $TotalMsgSize(i)$ for $i=1..900$. Finally, we divide $avgTotalMsgSize$ by the network size (n) in order to get the average number of bytes sent by one node $avgCommOverhead$. We use the last metric to measure the average energy spent by a node in communication.

Fig. 4.13 shows the impact of different simulation parameters on communication overhead. The effect of increasing cluster head probability (p) is shown in Fig. 4.13(a). We observe that the communication energy increase linearly as p increases. We can also notice that the rate increases significantly as the cluster radius (k) increases. This is can be clearly seen in Fig. 4.13(c) where it can be shown that the communication overhead is cubically proportional to the cluster radius (k). Mainly this cost is incurred during the JREQ phase as we will show analytically in section 4.6.4.

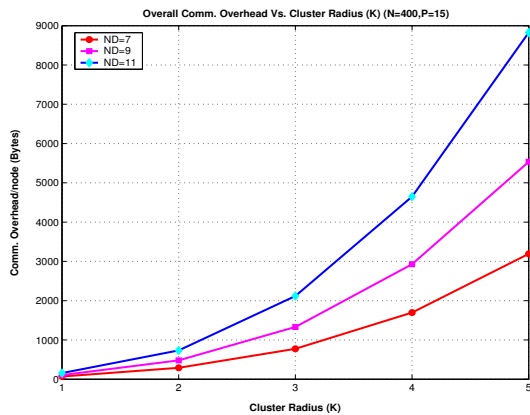
The effect of average node degree (d) on communication overhead is shown in



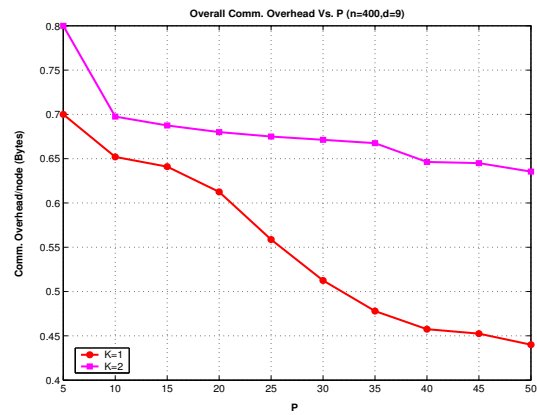
(a) The relation between communication overhead and the cluster head prob. (p)



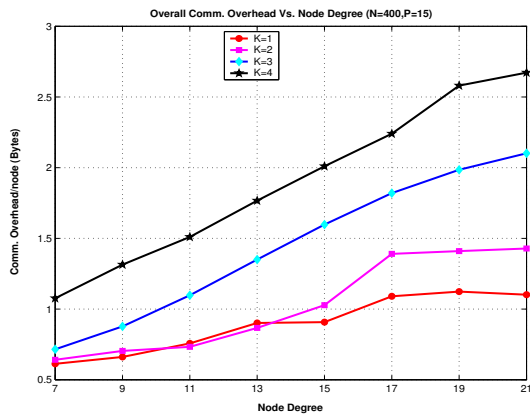
(b) The relation between communication overhead and the average node degree (d)



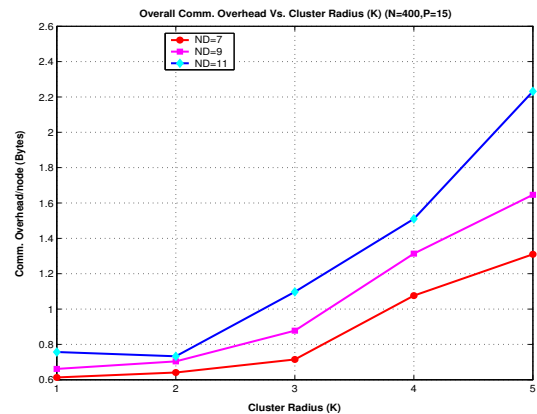
(c) The relation between communication overhead and the cluster radius (k)



(d) The standard deviation of the communication overhead as the cluster head prob. (p) increases



(e) The standard deviation of the communication overhead as the average node degree (d) increases



(f) The standard deviation of the communication overhead as the cluster radius (k) increases

Figure 4.13: The effect of different simulation parameters on communication overhead per node

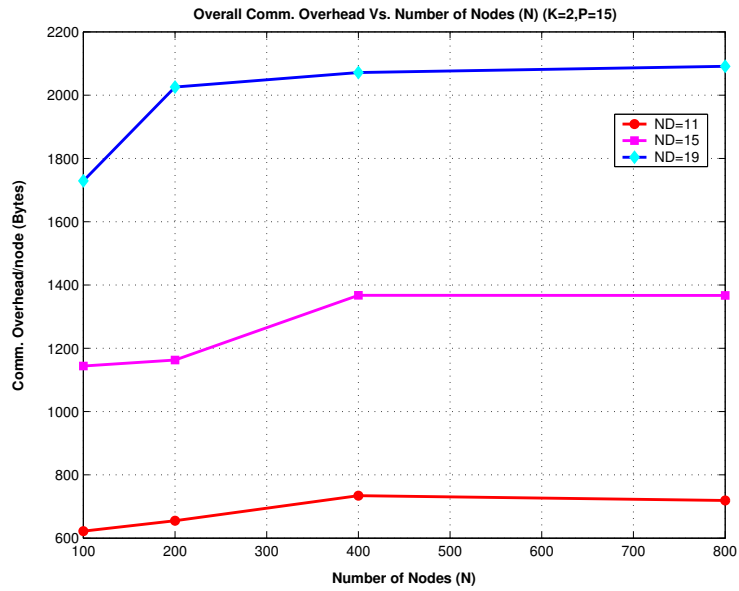
Fig. 4.13(b). We can notice that the communication overhead increases linearly as d increases. Although, we will discuss the relation between communication overhead and average node degree (d) analytically in section 4.6.4, we can intuitively explain that by analyzing the relation between average number of nodes per cluster N_c and average node degree (d) (Fig. 4.11(a)). As the average node degree increases, the average number of nodes per cluster increases linearly and hence the average number of JREQ messages increases linearly leading to a linear increase in the overall communication overhead.

Finally, we will show that OK is scalable in terms of processing time in section 4.7, (*lemma* 4.2). However, in this section, we study the scalability in terms of communication overhead. We tested the OK protocol for different network size ranging from 50 to 800 nodes. Fig. 4.14 shows the overall communication overhead per node as network size increases. We can clearly see that the number of bytes transmitted by a node slowly increases as the network size increases from 100 to 400. Then it remains almost constant afterwards. The standard deviation curves (Fig. 4.14(b)) show that this happens with high probability (± 2 bytes).

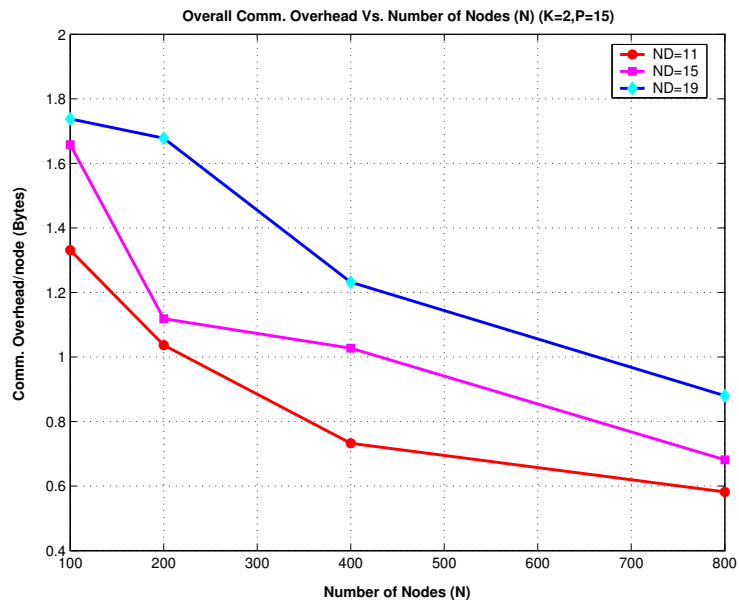
4.6 Analysis of The Results

In this section, using unit disk graph properties, and simple geometry, we will analytically show the following:

- The average number of nodes per cluster (N_c) is linear in d and quadratic in k (section 4.6.2).
- The average number of edges per cluster (E_c) is quadratic in both d and k (sec-



(a) The impact of network size (n) on the communication overhead incurred per node



(b) The standard deviation of the communication overhead as the network size n increases

Figure 4.14: Increasing the network size n does not effect the communication overhead

tion 4.6.2).

- The cluster head probability (p) does not affect the average overlapping degree (AOD) between clusters (section 4.6.3).
- The average overlapping degree (AOD) is linearly proportional to the average node degree (d) and quadratically proportional to the cluster radius (k) (section 4.6.3).
- The overall communication overhead is linearly proportional with d and cubically proportional with k (section 4.6.4).

We will start by describing the assumptions behind the proposed analytical model.

4.6.1 Assumptions

In order to simplify the proofs, we make the following assumptions:

- Each cluster can be approximated ideally by a circle of radius R .
- Since the transmission range of each node is fixed (T_r), and since only nodes that are within k hops from the cluster head can belong to this cluster, then we can approximate R as follows:

$$R = kT_r \tag{4.7}$$

In this case, R is considered the maximum euclidian distance that a node can be away from cluster head. Hence, the circle representing the cluster is considered the largest area that can be covered by a cluster.

- The cluster head is located at the center of this circle.

The above geometric representation of a k -hop cluster is considered the largest possible area for the cluster. This will lead to considering some areas as belonging to the cluster when they are not. We will refer to such an area as a *false area*. For example, if the cluster head node is located within distance R from the boundary of the sensor field, the circle representing the cluster will be clipped by the rectangle representing the field as shown in Fig. 4.15. Hence, the area which lies outside the sensor field is a *false area* since it is considered within the cluster but it does not really belong to it. Since the

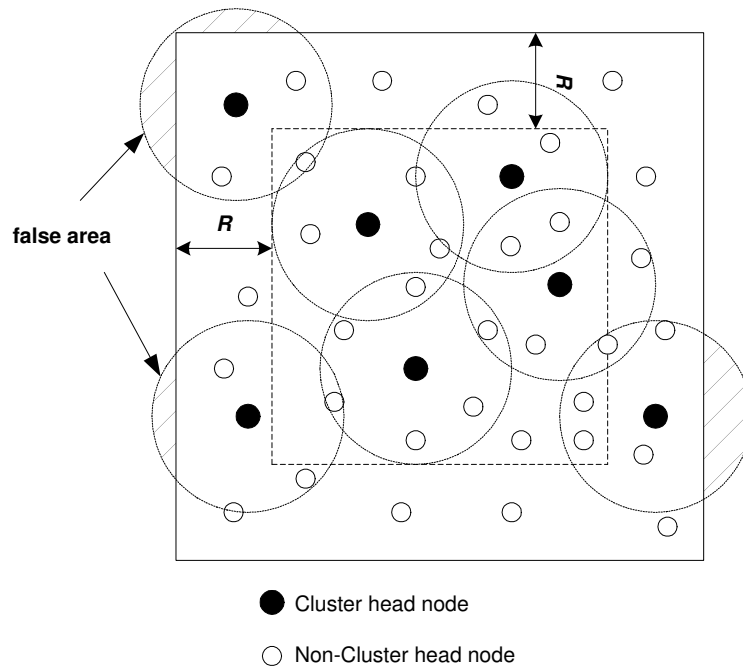


Figure 4.15: Circle representation of clusters

proposed analytical model represents an upper bound, *false area* will just make the upper bound not tight enough when compared with the simulation results. We will also show in the following that as the number of cluster heads increases, either because the network size (n) increases or the cluster head probability (p) increases, the probability of having

cluster heads within distance R decreases; hence the effect of *false area* decreases. So by carefully selecting the simulation parameters, we can safely ignore the effect of *false area*.

Recalling that the average number of cluster heads is pn , and assuming a square field with side length l , then the probability (P_{IN}) that a cluster head node is *at least* at distance R away from the boundary of the field (i.e. inside the dotted rectangle as shown in Fig. 4.15), is given by the following:

$$P_{IN} = \frac{(l - 2R)^2}{l^2} \quad (4.8)$$

Then the probability that a cluster head node is within distance R from the boundary will be (P_{OUT}):

$$P_{OUT} = 1 - P_{IN}$$

Let I be a discrete random variable representing the number of cluster heads that are within distance R from the boundary of the field. Then I can be expressed as a binomial distribution:

$$P(I = m|pn) = P_{OUT}^m P_{IN}^{pn-m} \binom{pn}{m}$$

and the expected number of cluster head nodes that are within distance R from the field boundary is:

$$E(I) = pnP_{OUT}$$

In order to ignore the effect of cluster heads that are near the boundary; hence decreasing the size of *false area*:

$$P_{OUT} \ll P_{IN}$$

$$\frac{l^2}{(l-2R)^2} \ll 2$$

$$4R^2 - 8lR + l^2 \gg 0$$

Solving the quadratic equations in R , and substituting $R = kT_r$, one of the following conditions must hold:

$$T_r \ll \frac{(1 - \sqrt{3}/2)l}{k} \quad (4.9)$$

OR

$$T_r \gg \frac{(1 + \sqrt{3}/2)l}{k} \quad (4.10)$$

The first condition was used in the simulations since it implies reducing the node transmission range; hence reducing energy consumption. However, we must be careful in reducing the node transmission range (T_r) since there is a minimum critical value for T_r in order for the graph to be connected [61]. Moreover, as k increases, it becomes difficult to satisfy the first condition while guaranteeing connectivity. Since our main goal is to have a connected graph, we have to violate the first condition as k increases. In a similar way, we notice that in order to increase the average node degree (d), we have to increase T_r ; hence, we may violate the first condition to achieve a certain average node degree. That's why we will notice that the analytical model diverges a little bit from the simulation results as k or d increases since the effect of cluster heads near the boundary starts increasing.

4.6.2 Average Cluster Size

We shall start by estimating an upper bound of the average cluster size (average number of nodes per cluster). The cluster will be represented by a circle with radius $R = kT_r$ as discussed in section 4.6.1. Assume that N_c is a discrete random variable representing the cluster size. Then using the same analysis as we did in the previous section, we can show that N_c can be expressed by the following binomial distribution:

$$P(N_c = m) = P_c^m (1 - P_c)^{n-m} \binom{n}{m} \quad (4.11)$$

where n is the network size and P_c is the probability that a node is inside the circle representing the cluster

$$P_c = \frac{\pi R^2}{l^2} = \frac{\pi k^2 T_r^2}{l^2} \quad (4.12)$$

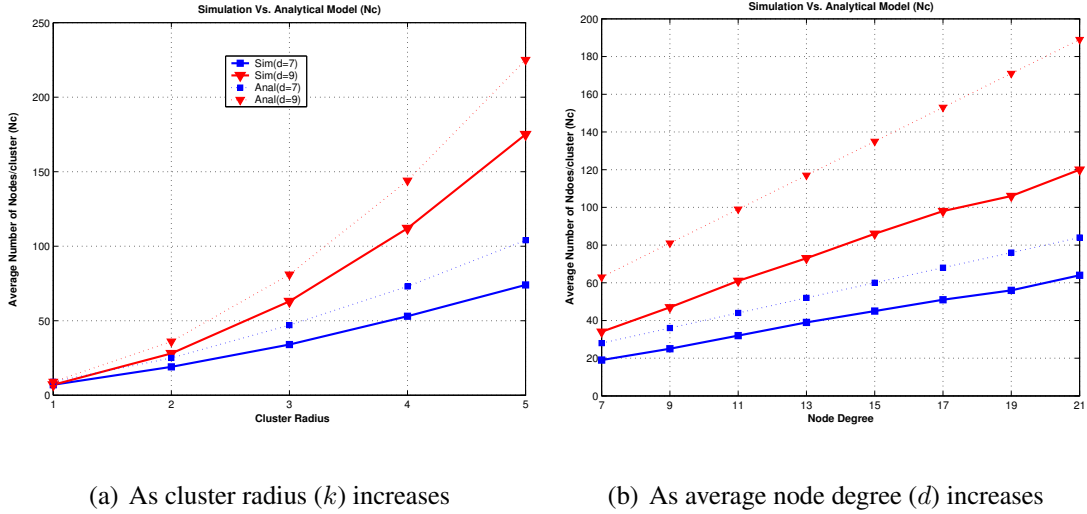
where l is the side length of the square field. Now substituting from equation 4.5, in order to get P_c in terms of average node degree (d), we get

$$P_c = \frac{dk^2}{n} \quad (4.13)$$

Hence the average cluster size ($E(N_c)$) is:

$$E(N_c) = nP_c = dk^2 \quad (4.14)$$

The above equation shows that the average cluster size is linearly proportional with average node degree (d) and quadratically proportional to the cluster radius (k). This conforms with the simulation results shown in section 4.6.2. Moreover, we can see that N_c is not a function of the cluster head probability (p). Fig. 4.16 shows the relation between the simulation results and analytical model given by equation 4.14.



(a) As cluster radius (k) increases (b) As average node degree (d) increases

Figure 4.16: The relation between the analytical model for average cluster size (N_c) and simulation results

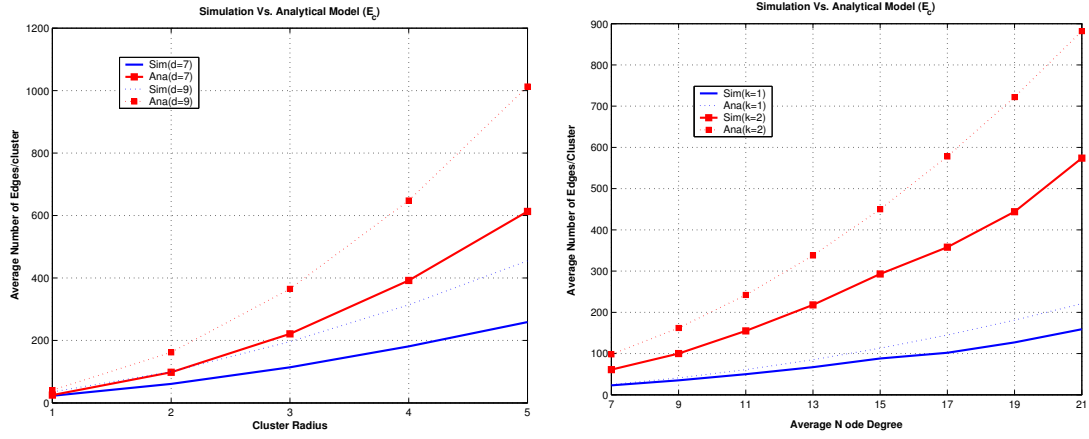
Using the above model, we can estimate the average number of edges per cluster (E_c) as follows. Since each node has an average node degree (d), and since the average number of nodes per cluster is N_c , then

$$E_c = \frac{dN_c}{2} = \frac{d^2k^2}{2} = O(d^2k^2) \quad (4.15)$$

Fig. 4.17 shows the relation between the simulation results and analytical model given by equation 4.15.

4.6.3 Average Overlapping Degree

Using the assumptions in section 4.6.1, we shall calculate an upper bound for the average overlapping degree (AOD). Assume that A, B are any two cluster head (CH) nodes. Then we recall from section 4.5 that the overlapping degree between the two corresponding clusters (\mathcal{O}) is a random variable where $\mathcal{O} = |N_k[A] \cap N_k[B]|$ and $N_k[A] \cap N_k[B] \neq \emptyset$. Notice that the overlapping degree is defined only for overlapping clusters (i.e. the



(a) As cluster radius (k) increases

(b) As average node degree (d) increases

Figure 4.17: The relation between the analytical model for average number of edges per cluster (E_c) and simulation results

random variable \mathcal{O} does not take the value 0). We define AOD as the mean of this random variable \mathcal{O} (i.e. $AOD = E(\mathcal{O})$). As shown in Fig. 4.18, the two clusters A and B are

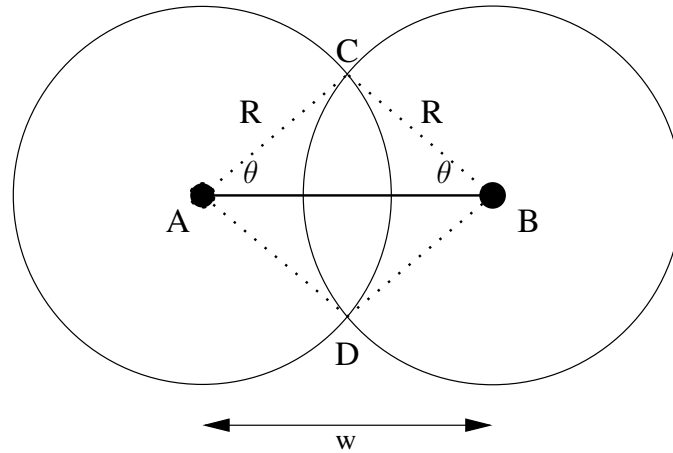


Figure 4.18: Overlapping Degree (\mathcal{O}) between two overlapping clusters

represented by two symmetric circles of radius $R = kT_r$. Instead of calculating the exact intersection of the two sets ($N_k[A] \cap N_k[B]$), we shall estimate the intersection of the two sets by the area of intersection between the two corresponding circles. Let W be the

euclidian distance between the two CH nodes. Then, W is a continuous random variable that can take values ranging from 0 to $2R$. The two clusters are completely overlapped if $W = 0$ and there is no overlapping if the distance between the two cluster heads is greater than or equal $2R$. Let $F(w)$ and $f(w)$ be the CDF and PDF of the random variable W consequently. Then

$$F(w) = P(W < w) = \frac{\pi w^2}{\pi(2R)^2} = \frac{w^2}{4R^2} \quad (4.16)$$

$$\therefore f(w) = \frac{dF(w)}{dw} = \frac{w}{2R^2} \quad (4.17)$$

We will express \mathbf{O} as a function of w as follows. The area of intersection between two symmetric circles A and B (I_{AB}) is ³:

$$I_{AB} = (2\theta - \sin 2\theta)R^2 = E(\mathbf{O} | w) \quad (4.18)$$

where $w = 2R \cos \theta$ (using cosine rule). Hence, \mathbf{O} is a continuous random variable that is represented as a function of θ or w alternatively.

$$\therefore E(\mathbf{O}) = \int_0^{2R} E(\mathbf{O} | w) f(w) dw = \int_0^{2R} (2\theta - \sin 2\theta) R^2 f(w) dw \quad (4.19)$$

Substituting from Eq. 4.17, 4.18, and noticing that $dw = 2R \sin \theta$, we have the following:

$$E(\mathbf{O}) = \int_0^{\pi/2} (2\theta - \sin 2\theta) R^2 2 \sin \theta \cos \theta d\theta = R^2 \int_0^{\pi/2} (2\theta - \sin 2\theta) \sin 2\theta d\theta \quad (4.20)$$

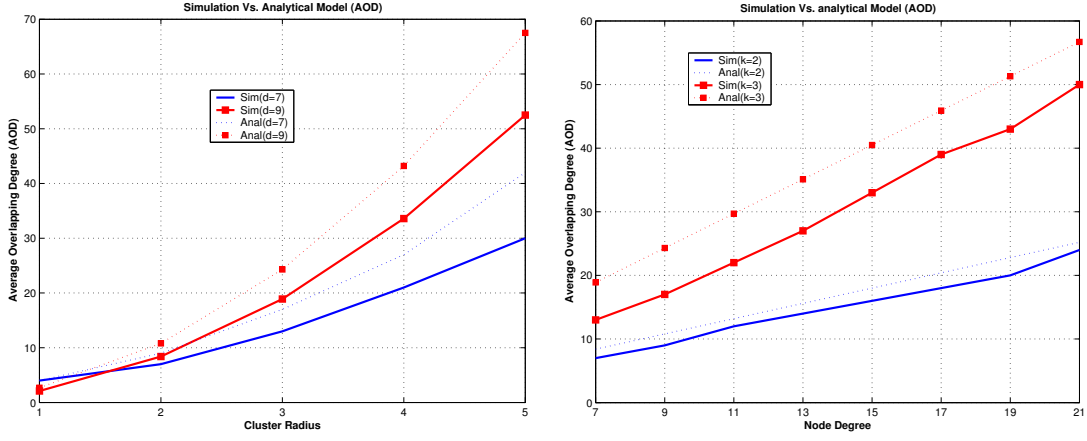
It can be shown that $\int_0^{\pi/2} (2\theta - \sin 2\theta) \sin 2\theta d\theta = \frac{\pi}{4}$. Hence,

$$E(\mathbf{O}) = \frac{\pi R^2}{4} \quad (4.21)$$

Substituting from Eq. 4.7, we have:

$$E(\mathbf{O}) = \frac{\pi k^2 T_r^2}{4} \quad (4.22)$$

³For more details of the proof, please refer to appendix A



(a) As cluster radius (k) increases

(b) As average node degree (d) increases

Figure 4.19: The relation between the analytical model for average overlapping degree (AOD) and simulation results

Substituting from Eq. 4.5 to get the relation in terms of average node degree, we reach the following:

$$E(\mathcal{O}) = \frac{dk^2l^2}{4n} = \frac{dk^2}{4\mu} = \mathbf{AOD} \quad (4.23)$$

The above equation shows that the average overlapping degree is linearly proportional with average node degree d and quadratically proportional to the cluster radius k . This conforms with the simulation results shown in section 4.6.3. We can also notice that AOD is not a function of cluster head probability p . Fig. 4.19 shows the relation between the simulation results and analytical model given by equation 4.23.

4.6.4 Overall Communication Overhead

In this section, we will calculate an upper bound of the average number of control messages transmitted by a node. As we did in the previous sections, the cluster will be approximated by a circle with radius $R = kT_r$. Recall that there are two phases in the

OK protocol: the cluster head advertisement phase (CHAD phase) and the join request phase (JREQ phase). We will estimate the number of messages sent during each phase per node. Then the overall communication overhead per node will be the total number of messages. We will start by estimating the average number of nodes that are exactly k hops away from the cluster head (n_k). From equation 4.14, the average number of nodes in k -hop cluster is:

$$E(N_c) = nP_c = dk^2 = E_k(N_c) \quad (4.24)$$

Then

$$n_k = E_k(N_c) - E_{k-1}(N_c) = dk^2 - d(k-1)^2 \quad (4.25)$$

$$\therefore n_k = d(2k-1) \quad (4.26)$$

Using the above results, we can calculate the average number of CH_AD messages sent during the CHAD phase. Initially, the cluster head (CH) node broadcasts one CH_AD message to neighbors. The message is then flooded for k hops with no duplication (i.e. if a node received the same CH_AD message multiple times, it will just forward it once to its neighbors. Hence, the CH_AD message is forwarded through the edges of a spanning tree of the cluster graph as shown in Fig. 4.20. Initially, the CH node broadcasts one message to all its neighbors $\{A, C, D\}$. Now each of those nodes will broadcast the same message to its corresponding neighbors, after incrementing the hop count. Hence, B will receive the same message from $\{A, C\}$ and N will receive the same message from $\{C, D\}$. However, since the OK protocol uses smart flooding, the second CH_AD will be dropped by both B and N. The CH_AD broadcast will continue for k hops away from the CH node (in this particular example, $k = 5$).

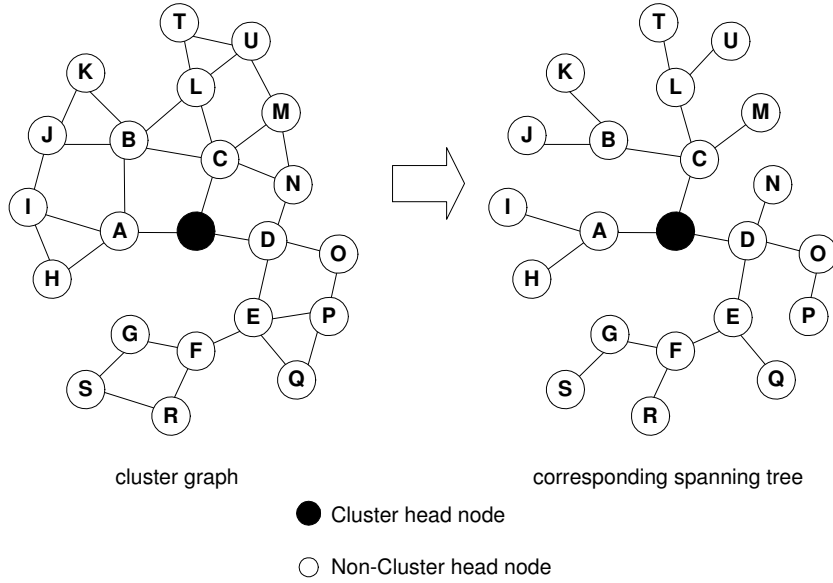


Figure 4.20: The CH_AD message will follow a spanning tree rooted at the CH node ($k = 5$)

Let M_{CHAD} be the average number of CH_AD messages broadcasted within the cluster. Then M_{CHAD} is equal to the average number of non-leaf nodes in breadth-first tree of the graph rooted at the CH node.

$$M_{CHAD} = 1 + \sum_{i=1}^{k-1} n_i \quad (4.27)$$

where n_i is the expected number of nodes that are exactly i hops way from the CH node (Eq. 4.26). Substituting from Eq. 4.26 and simplifying the expression, we reach the following:

$$M_{CHAD} = 1 + \frac{2d(k-1)^2}{2} = O(dk^2) \quad (4.28)$$

Using a similar approach, we can calculate the average number of JREQ messages (M_{JREQ}) unicast from non-CH nodes to the CH node. We assume that we do not do

any aggregation of the messages⁴; hence; a JREQ message, unicasted from a leaf node in the spanning tree, will be forwarded k times till it reach the CH node. Therefore, M_{JREQ} can be calculated as follows:

$$M_{JREQ} = kn_k + (k - 1)n_{k-1} + \dots + 2n_2 + n_1 = \sum_{i=1}^k in_i \quad (4.29)$$

Substituting from Eq. 4.26 and simplifying the expression, we reach the following expression:

$$M_{JREQ} = \frac{dk(4k - 1)(k + 1)}{6} = O(dk^3) \quad (4.30)$$

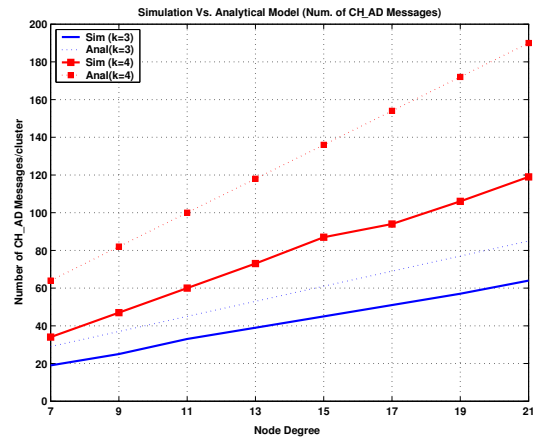
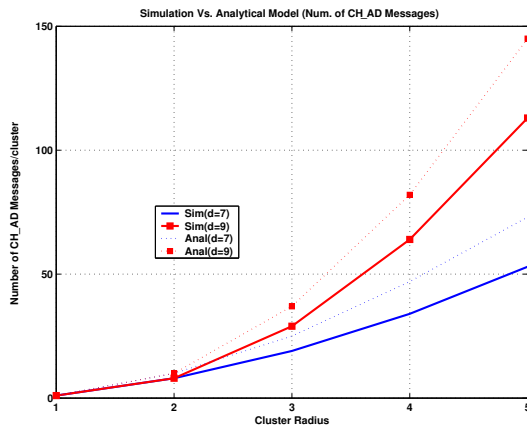
Fig. 4.21 shows the relation between the simulation results and analytical model of the communication overhead.

4.7 Correctness and Complexity

In this section we shall discuss that the OK protocol provided in Fig. 4.4 meets the following design goals (requirements):

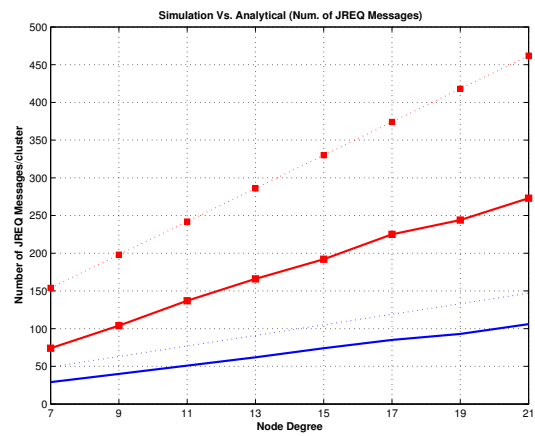
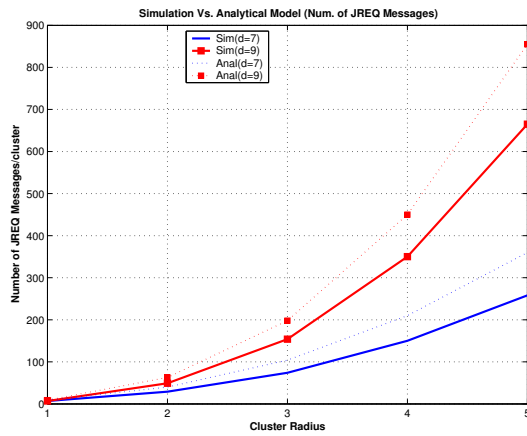
1. Completely distributed.
2. Terminates within $O(k)$ iterations, regardless of network diameter, where k is the cluster radius.
3. At the end of the algorithm, each node is either a cluster head, or non-cluster head node that belongs to one or more clusters.

⁴Of course, if message aggregation is used, the overall communication overhead will improve. So the above analysis is considered a worst case analysis.



(a) Average number of CH_AD messages per cluster as cluster radius (k) increases

(b) Average number of CH_AD messages per cluster as average node degree (d) increases



(c) Average number of JREQ messages per cluster as cluster radius (k) increases

(d) Average number of JREQ messages per cluster as average node degree (d) increases

Figure 4.21: The relation between the analytical model for overall communication overhead per node and simulation results

4. Efficient in terms of memory usage.

Observation 1. OK is completely distributed (requirement 1). A node can either elect to become a cluster head, or join a cluster if it receives CH_AD messages within its cluster radius. Thus, node decisions are based solely on local information.

Lemma 4.1. *The time complexity of OK is $O(k)$ (requirement 2).*

Proof. The worst case scenario is: a non-CH (NCH) node does not receive any CH_AD messages and changes its status to CH. Then broadcasts a CH_AD message and waits for JREQ messages. Recall from section 4.4.3 that the maximum time that an NCH node waits for a CH_AD message is equal to $t(k) + \delta$, where $t(k)$ is the time needed for a message to travel k hops and δ is a constant value independent from k . Hence, the total time of this worst case scenario is $t(k) + \delta + 2t(k)$. Therefore the maximum time that a node should wait before terminating OK is $t(k) + \delta + 2t(k) = 3t(k) + \delta = O(k)$. \square

Lemma 4.2. *At the end of the OK algorithm, a node is either a cluster head, or non-cluster head node that belongs to one or more clusters (requirement 3).*

Proof. Initially each node is either CH or NCH node. If the node is a CH node, it will terminate the OK algorithm after $2t(k) + \delta$ time units when the JREQ_WAIT timer fires. In case of NCH node, after $t(k) + \delta$ time units, either it joins one or more clusters that it heard from or changes status to CH and terminates the OK algorithm after $2t(k)$ time units. \square

Lemma 4.3. *The expected number of adjacent overlapping clusters is $O(pdk^2)$, where p is the cluster head probability, d is the average node degree, and k is the cluster radius.*

Proof. Recall that the expected number of clusters is np where n is the network size. Let u and v be two cluster head nodes. Then the two corresponding clusters of u and v are overlapping iff $dist_G(u, v) < 2k$. Using the circle approximation of the cluster as discussed in section 4.6.1, then the probability (P_{Adj}) that a CH node is within distance $2R$, $R = kT_r$, from m other CH nodes is given by the following binomial distribution:

$$P_{Adj}(m) = P_{2R}^m (1 - P_{2R})^{np-m-1} \binom{np-1}{m}, \text{ where } P_{2R} = \frac{\pi(2R)^2}{l^2} \quad (4.31)$$

Hence, the expected number of adjacent clusters is ($E(P_{Adj})$):

$$E(P_{Adj}) = P_{2R}(np-1) \simeq npP_{2R} = \frac{4\pi npR^2}{l^2} \quad (4.32)$$

Since $R = kT_r$, substituting from equation 4.5 and simplifying the expression, we get the following:

$$E(P_{Adj}) = 4\pi pdk^2 = O(pdk^2) \quad (4.33)$$

□

Lemma 4.4. *The OK algorithm has an average memory usage of $O(1)$ per node (requirement 4).*

Proof. The two major data structures used by the OK protocol are: *CH_table* and *AC_table*. Any other data structures will take $O(1)$ memory to store. Recall from section 4.4.1, *CH_table* is used by each node, whether CH or NCH, to store information about the known CH nodes. Hence, the average size of the *CH_table* is equal to the expected number of clusters that cover a certain node; which is equal to the expected number of adjacent clusters ($E(P_{Adj})$). Therefore, using lemma 4.3, the average size of the *CH_table* is $O(dk^2)$.

Since both d , and k are constants and independent of the network size, the average size of CH_table is $O(1)$ ⁵.

Recall from section 4.4.1, AC_table is used by only CH nodes to keep track of adjacent clusters. Hence, we can calculate the average size of AC_table as follows:

$$\text{size}(AC_table) = E(P_{Adj}) \times \text{the expected number of boundary nodes}$$

However, the expected number of boundary nodes is equal to the average overlapping degree (AOD). Substituting from Eq.4.23, we get the following:

$$\text{size}(AC_table) = E(P_{Adj}) \times \frac{dk^2}{4\mu} = O\left(\frac{d^2k^4}{\mu}\right)$$

Since both d , and k are constants and independent of the network size, the average size of AC_table is $O(1)$. Hence, on the average, the total memory usage per node is $O(1)$. \square

⁵Notice that the maximum size of CH_table can not exceed the average number of clusters (pn)

Chapter 5

The Local Location Discovery Phase

The focus of this chapter is on estimating the relative nodes' positions within local cluster. We will start by formulating the problem and show how we can select a local coordinate system (LCS) for the cluster. Then we will discuss the major cause of error (reflection error) and propose some heuristics to reduce this error. We then propose the Multi-hop Relative Location Estimation (MRLE) algorithm and the following refinement step as an optional enhancement to the estimated position. Finally the accuracy of the proposed scheme is evaluated through simulation. The results confirm the high accuracy of the positions estimated by our approach and capture the impact of the different parameters, such as cluster size (n_c), cluster radius (k), and connectivity (d) on the accuracy of the estimated position. We also introduce a new metric, the CLIQUE factor (CF), as a measure of performance. The CLIQUE factor of a cluster measures how close the subgraph induced by cluster to a complete graph. We will show that the CF is the major factor affecting accuracy regardless of cluster size.

5.1 Problem Definition

The local location discovery (LLD) problem can be formalized as follows:

”Given a subset of ad-hoc network where each node knows the distance measurements, perhaps with some high margin of error, between nodes that are within its listening range, the objective is to construct a local map for this cluster with accurate relative node positions such that the error between estimated distances based on the local map and measured distances is minimized.”

At the end of the cluster formation phase, the cluster head node receives node-to-node range measurements for each node belonging to this cluster. This information is stored in the local cluster graph (LCG) data structure, as described in the previous chapter. During the *LLD* phase the cluster head node uses the LCG to build a local coordinate system for its corresponding cluster as follows.

Let (x_i, y_i, z_i) be the 3-D position of node i , where $i = 0, 1, \dots, n_c - 1$, and n_c is the number of nodes in the cluster including the cluster head node. Let P be an $(n_c \times 3)$ matrix representing all node positions such that:

$$P = \begin{bmatrix} x_0 & y_0 & z_0 \\ x_1 & y_1 & z_1 \\ \vdots & \vdots & \vdots \\ x_{n_c-1} & y_{n_c-1} & z_{n_c-1} \end{bmatrix} \quad (5.1)$$

Let $P(i)$ be a function that returns the position of node i such that: $P(i) = [x_i, y_i, z_i]$. Assume that D is an $(n_c \times n_c)$ matrix representing the intra-node distance measurements *as estimated by the nodes* during the cluster formation phase. Keep in your mind that these measurements could have an error. Let $D_p = D(P)$ be a vector function that returns an $(n_c \times n_c)$ matrix representing the *calculated* intra-node distances given a node position estimate P , where

$$D_P[i, j] = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}, i, j = 0, 1, \dots, n_c - 1 \quad (5.2)$$

Given a certain position estimate P , define the following vector error function E :

$$E(P) = \sum_{i=1}^{n_c-1} \sum_{j=0}^{i-1} (D_P(i, j) - D(i, j))^2 \quad (5.3)$$

The error function measures the least square error between the estimated inter-node distances (D_P) using P and the actual measured distances measured by nodes (D). The objective is to find the *optimal node positions* P^* that minimizes the error function E .

In the LLD phase, we first start by estimating an initial relative position (P_0) for all the nodes within the cluster. We introduce the Multi-hop Relative Location Estimation (MRLE) algorithm. MRLE uses inter-node distances in order to estimate relative node position with respect to some local coordinate system (LCS). Then we introduce an optional refinement step, where we iteratively improve the initial position estimate such that the error function E (Eq. 5.3) is minimized. Compared with the MREL algorithm, the re-fit step is much more expensive, in terms of computation power. Hence, it is optional if we are seeking higher accuracy. We will see in the results section that the accuracy of the estimated initial position (P_0) using MRLE is acceptable. For simplicity, we shall use

2-D coordinates in the following analysis. However, the technique can be easily extended to the 3-D case.

5.2 The Local Coordinate System (LCS)

The relative node position is calculated with respect to some coordinate system. We shall refer to this coordinate system as the local coordinate system (LCS). The local coordinate system (LCS) is defined by fixing three non-colinear nodes: R_0 , R_1 , and R_2 . We shall refer to these nodes as the *reference nodes*. For simplicity, we will assume that the cluster head node is the origin of the coordinate system (R_0). However, we will discuss later in this section that this is not always the case and any other node could be the origin.

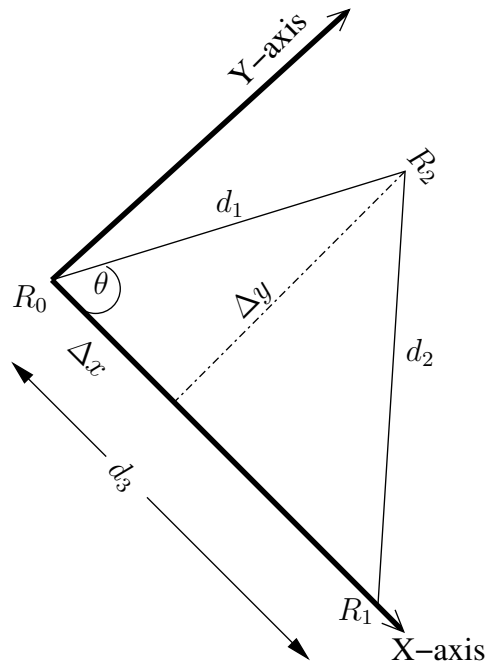


Figure 5.1: The local coordinate system (LCS)

To form the positive x -axis, we select another node R_1 that is within the transmission range of R_0 as shown in Fig. 5.1. Finally, a third node that is within the transmission

range of both R_0 and R_1 , but not *co-linear* with them, will be positioned in the upper half-plane. Hence, the y -axis is selected to be perpendicular to the x -axis in the direction of R_2 as shown in Fig. 5.1. Thus, the placement of R_1 has the effect of fixing a particular rotational orientation, while the placement of R_2 locks in a particular reflective orientation. The positions of the nodes R_0 , R_1 , and R_2 are given as follows:

$$\begin{aligned} P_0(R_0) &= (0, 0) \\ P_0(R_1) &= (D(R_0, R_1), 0) \end{aligned} \tag{5.4}$$

The coordinates of $R_2 = (x_2, y_2)$ can be computed using the law of cosines as follows:

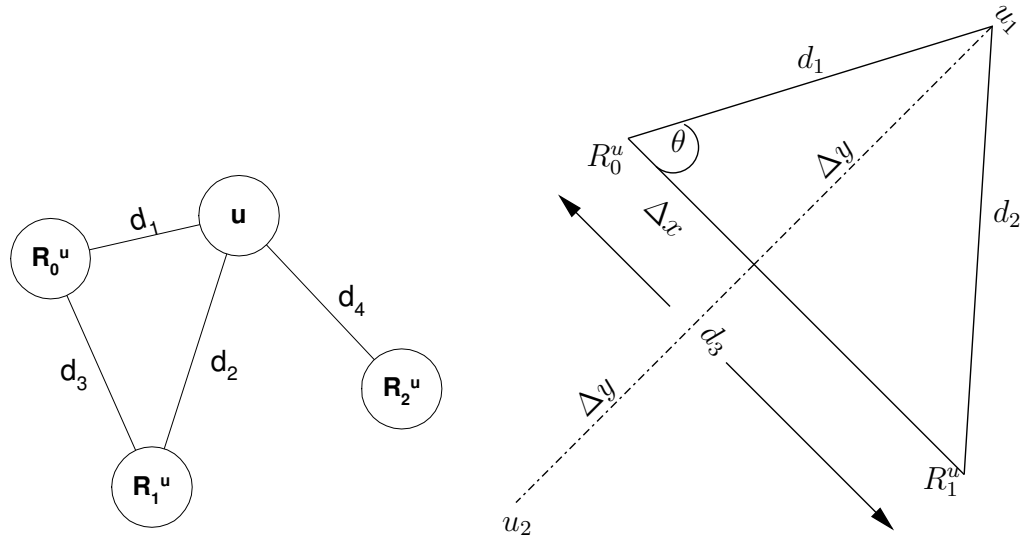
$$\begin{aligned} \cos(\theta) &= \frac{d_1^2 + d_3^2 - d_2^2}{2d_1d_3} \\ x_2 &= d_1 \cos(\theta) \\ y_2 &= \sqrt{d_1^2 - x_2^2} \end{aligned} \tag{5.5}$$

where $d_1 = D(R_0, R_2)$, $d_2 = D(R_1, R_2)$, $d_3 = D(R_0, R_1)$ and D is the distance matrix representing the measured distances by nodes during bootstrapping. After selecting the LCS, we compute the relative node positions starting from the neighbors of the reference nodes first and moving away from the LCS towards the border of the cluster. We will discuss this in more details in the next three sections.

5.3 Relative Position Estimation Using Three Distances

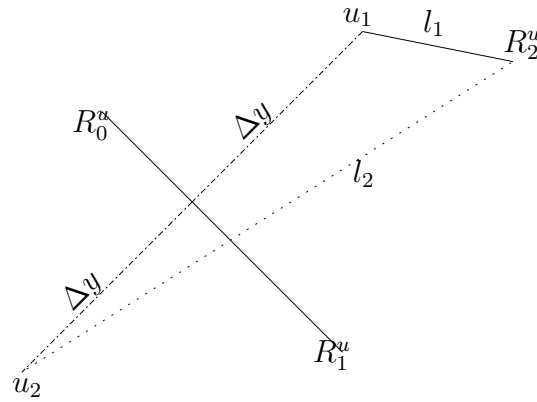
Assume that we want to estimate the position of a node u that knows the distances to three *non-colinear* neighbors R_0^u , R_1^u , and R_2^u as shown in Fig. 5.2(a). The distances were estimated during the network bootstrapping as discussed in the previous chapter. We will further assume that we know the relative position of the three nodes.

Let $d_1 = D(R_0^u, u)$ and $d_2 = D(R_1^u, u)$ where D is the distance matrix representing the measured distances by nodes during bootstrapping. Let $\overrightarrow{R_0^u R_1^u}$ be the vector connecting R_0^u and R_1^u in the direction of R_1^u as shown in Fig. 5.2. We will refer to the vector



(a) Node u knows the distance to three neighbors

(b) Node u has two candidate positions



(c) selecting one candidate

Figure 5.2: Estimating the position of a node (u) using three distances

$\overrightarrow{R_0^u R_1^u}$ as the *base line*. Let $d_3 = \|\overrightarrow{R_0^u R_1^u}\|$ be the length of this vector. Let \vec{x} be the unit vector in the direction of the *base line* ($\overrightarrow{R_0^u R_1^u}$). Let \vec{y} be the unit vector in the direction perpendicular to the *base line* in the direction¹ of R_2^u . Then there are two candidate positions for the node u (u_1 and u_2) that can be computed, using vector notations, as follows:

$$\begin{aligned}\cos(\theta) &= \frac{(d_1^2 + d_3^2 - d_2^2)}{(2d_1 d_3)} \\ \Delta_x &= d_1 \cos(\theta) \\ \Delta_y &= \sqrt{(d_1^2 - \Delta_x^2)} \\ \vec{u}_1 &= \vec{R}_0^u + \Delta_x \vec{x} + \Delta_y \vec{y} = (u_x, u_y) \\ \vec{u}_2 &= \vec{R}_0^u + \Delta_x \vec{x} - \Delta_y \vec{y} = (u_x, -u_y)\end{aligned}$$

Notice that u_2 is the reflection of u_1 across the base line. The third node R_2^u is used to resolve the reflection as shown in Fig. 5.2(c). Let $d_4 = D(u, R_2^u)$ (i.e. the measured distance between the two nodes during network bootstrapping). We chose the candidate that is closer to R_2^u as compared to d_4 . The node R_2^u is called the *reflection resolver*. So in the example given in Fig. 5.2(c), we will chose u_1 since $|d_4 - l_1| < |d_4 - l_2|$. Resolving reflection is the most important decision in relative position estimation. We will discuss the problem in more details in the next section and show that we can not always resolve the reflection.

5.4 Multi-hop Relative Position Estimation

As discussed in section 5.2, the local coordinate system (LCS) is defined by three reference nodes: R_0, R_1 , and R_2 . Initially, we fix the position of the three reference nodes as shown before. We will refer to the set of nodes that have known relative position as the set of *Identified Nodes* (I). It follows that, initially, $I = \{R_0, R_1, R_2\}$. Using the

¹It does not matter which direction \vec{y} is pointing too as long as it is perpendicular to \vec{x}

identified nodes, we can calculate the relative position of other nodes that are multi-hop away from the LCS. For example, using the subgraph shown in Fig. 5.3, since nodes A, B, C know distances to three *non-collinear* neighbors ($R_0, R_1, \text{and } R_2$), they can estimate their positions as described in section 5.3. Hence, the set of identified nodes now include $\{R_0, R_1, R_2, A, B, C\}$. Now we can estimate the position of nodes D and E since they have distances to three or more *identified nodes*. In a similar way, we can calculate the position of nodes F and G , then nodes H and I . One thing to notice here is that as we move away from the LCS, the error accumulation increases. We will describe some heuristics to limit the accumulated error section 5.7.

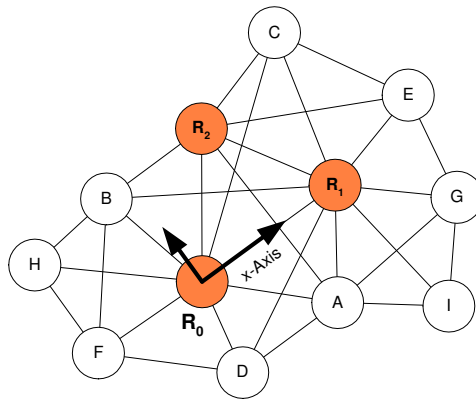


Figure 5.3: Propagation of node position estimating starting from the reference nodes and moving towards the border of the cluster.

5.5 Relative Position Estimation Using Two Distances

Now assume that the node u knows the distances only to two neighbors² R_0^u and R_1^u as shown in Fig. 5.4. Normally this will happen when the node is on the border of the cluster. Hence, we have already computed the relative positions of the nodes that are away from the border. Using the base line $\overrightarrow{R_0^u R_1^u}$, we can calculate two candidate positions for the node u as discussed in the previous section. In order to resolve the reflection and select one candidate, we will use a simple observation.

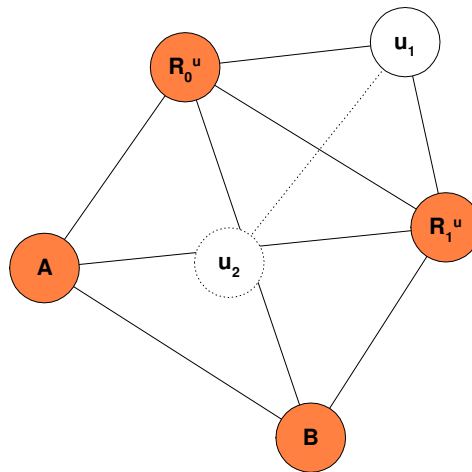


Figure 5.4: Estimating the position of a node (u) using two distances

Assume a node v with known relative position and $v \notin \{R_0^u, R_1^u\}$. If the distance between v and candidate $u_i, i = 1, 2$ is less than the node transmission range (T_r) then this contradicts with the fact that v is not a neighbor of u ; hence candidate position u_i can not be the right position. Applying this heuristic to the example shown in Fig. 5.4. The candidate position u_2 is within the transmission range of nodes A and B . Hence, the

²A similar situation is when the node u has three or more colinear neighbors; hence we can not resolve reflection.

correct position is u_1 .

5.6 The Reflection Error

In section 5.3 we discussed how the base line $(\overrightarrow{R_0 R_1^u})$ is used to calculate two candidate positions for a node u and how the *reflection resolver* (R_2^u) is used to select one candidate. Notice that if the wrong candidate is selected, the error in the position of the node is equal to $2\Delta y$, where Δy is the perpendicular distance between u and the base line as shown in Fig. 5.2. Actually, the reflection error in one node can cause much worse error: the *reflection propagation error*.

Consider the subgraph shown in Fig. 5.5(a). Initially the set of identified nodes (I) contains the three reference nodes. Using the base line $\overrightarrow{R_0 R_1}$, we can calculate the two candidate positions for node A of which one is wrong (A_e). Now using the node R_2 to resolve reflection, and noticing that the measured distances contain some error, we can select the wrong candidate for node A_e . Now the set of identified nodes contains $\{R_0, R_2, R_2, A_e\}$. We can now estimate the position of node B using $\overrightarrow{R_1 R_2}$ as the base line and node A_e as a reflection resolver. That will lead to estimating a wrong position for node B (B_e) as shown in Fig. 5.5(b). In a similar way, the reflection error will propagate to node C . What makes the problem worse is that the refinement phase can not correct this type of error.

The best way to solve this problem is to avoid it! So the question is what leads to a reflection error. The answer is: *a skinny triangle*. Fig. 5.6(a) shows an example when the reflection resolver (R_2^u) is close to the base line $(\overrightarrow{R_0 R_1^u})$. In this case, the difference

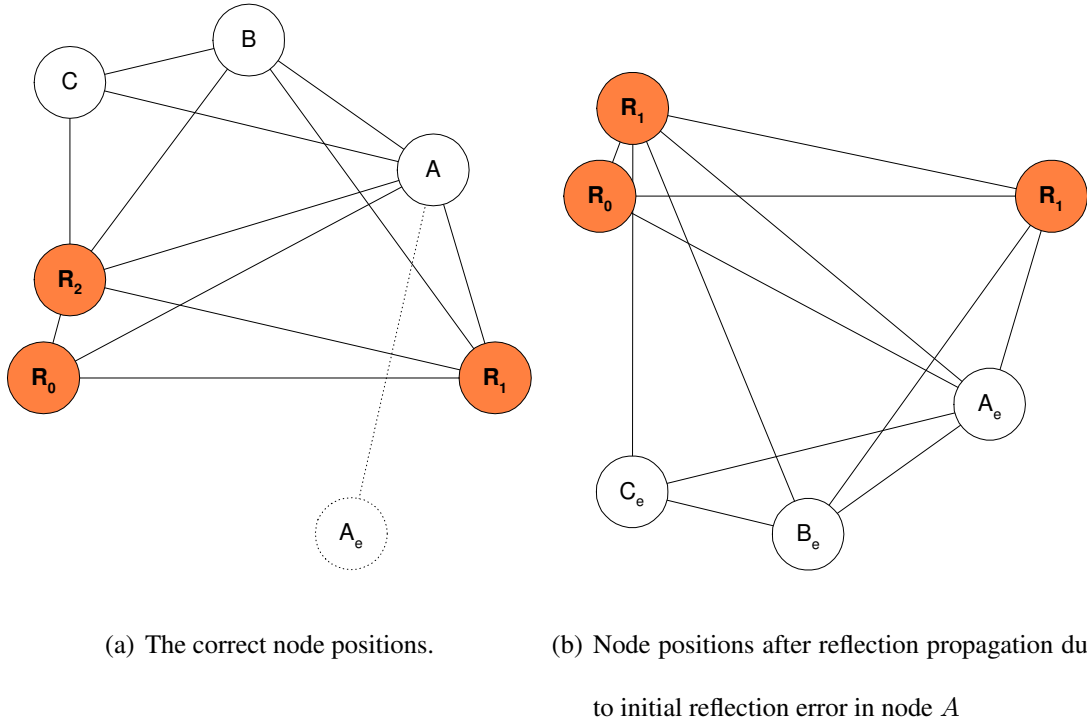


Figure 5.5: The reflection propagation phenomena.

between l_1 and l_2 is very small; hence, we may not be able to resolve reflection correctly. Another case of skinny triangle is shown in Fig. 5.6(b). Here the node u is close to the base line and again l_1 is very close to l_2 . From the two examples, we can notice that a skinny triangle will exist if $\| \overrightarrow{u_2 R_2^u} - \overrightarrow{u_1 R_2^u} \| \leq \delta$ where δ is some threshold depending on the error in the measured distances. In the simulation results, we set $\delta = 3\sigma$, where σ is the standard deviation of the range estimation error.

5.7 Heuristics to Limit Error Accumulation

In order to limit error accumulation, we will assign to each node an *error level* (EL). The error level of the node is value indicating how much error the node may have. We assume that the three *reference nodes* (R_0 , R_1 and R_2), representing the local coordinate system

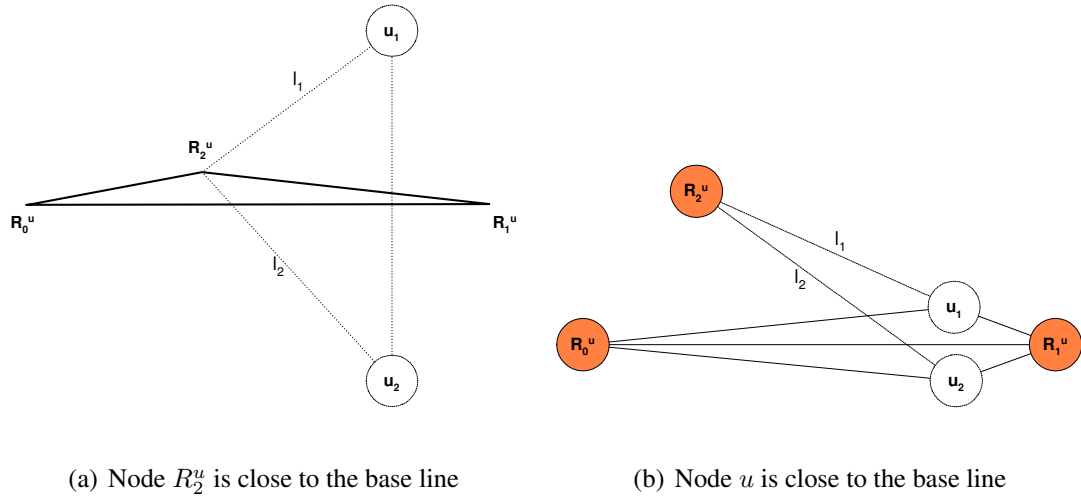


Figure 5.6: Different cases for skinny triangles

(LCS), have an error level of 0 (i.e. $EL(R_0) = EL(R_1) = EL(R_2) = 0$). Then the error level of a node u ($EL(u)$) can be computed as a function of the error levels of the the base-line nodes (R_0^u and R_1^u) as follows:

$$EL(u) = \frac{EL(R_0^u) + EL(R_1^u)}{2} + 1$$

As the node moves away from the LCS, its error level increases because of error accumulation. In the remainder of this section we will discuss two techniques to limit the error accumulation.

5.7.1 Selecting The Local Coordinate System (LCS)

As discussed in section 5.2, the local coordinate system (LCS) is defined by fixing three non-colinear nodes (the *reference nodes*): R_0 , R_1 , and R_2 . The three reference nodes together form a triangle $\Delta(R_0, R_1, R_2)$. We will refer to this triangle as the *LCS triangle*. The selection of this triangle affects the overall accuracy of the estimated node positions. As we showed in Fig. 5.5(a), if the LCS triangle is skinny this may lead to the *reflection*

propagation error. Moreover, since the set of *identified nodes* (I) initially contains only those three nodes, the error in the estimated position of all other nodes will be highly effected by the error in the estimated position of the reference nodes (specially R_1 and R_2). Finally, as we move away from the LCS, the node error level increases (i.e. the accumulated error increases). Based on this it is better to have the LCS near the center of the graph. That is why we assume that the cluster- head (CH) node is the origin (R_0) since, most probably, it will be at the center of the cluster. However, this is not always the case as shown in Fig. 5.7, where the CH node is on the border of the cluster. Usually this happens when the CH node is near the boundary of the sensor field.

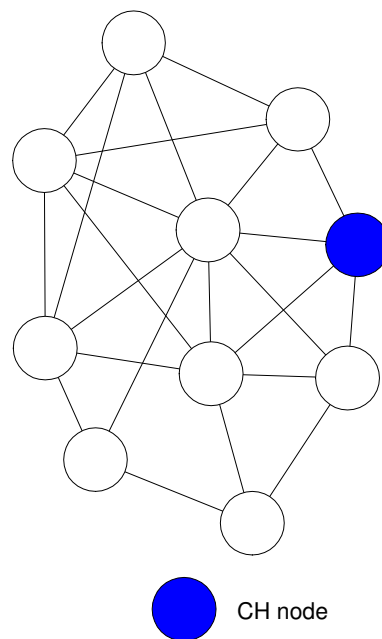


Figure 5.7: The cluster-head (CH) node is on the border of the cluster ($k=2$)

Based on the above analysis, the LCS should have the following features:

1. The LCS triangle must not be skinny. We shall use the aspect ratio (AR) of the LCS triangle to measure how skinny it is where the higher the aspect ratio, the skinnier

the LCS triangle and vice versa. Assuming the side lengths of the LCS triangle are: d_1, d_2 and d_3 . Then the aspect ratio (AR) of the triangle can be calculated as the ratio between the circum radius (CR) and inner radius (IR) of the triangle as follows:

$$\begin{aligned}
 s &= \frac{d_1+d_2+d_3}{2}; \\
 CR &= \frac{d_1*d_2*d_3}{4*\sqrt{s(s-d_1)(s-d_2)(s-d_3)}}; \\
 IR &= \sqrt{\frac{(s-d_1)(s-d_2)(s-d_3)}{s}}; \\
 AR &= CR/IR;
 \end{aligned} \tag{5.6}$$

2. The error in estimating the positions of R_1 and R_2 should be minimized. Notice that the position of R_1 is $(D(R_0, R_1), 0)$ (Eq. 5.4). Hence, the error in R_1 position depends on the range estimation technology used. For example, if we are using RSSI (section 2.1.2), then as the distance between R_0 and R_1 increases, the error in the distance increases. Hence, it is better to chose R_1 to be closer to R_0 , without violating the skinny triangle condition.
3. The LCS should be near the center of the local cluster graph. Most of the time, the CH node will be the origin of the LCS since it is almost the center of the cluster.

Assuming that there are n_c nodes in the cluster, then there is a maximum of $C_2^{n_c-1}$ different possible LCS triangles that can be formed, i.e. different local coordinate systems (LCS). Of course, the actual number is less than this since the three reference nodes must be *non-colinear neighbors*. In order to show the effect of the selection of the LCS on the accuracy, we propose four different heuristics to select the local coordinate system (LCS) as follows³:

³For simplicity, we will assume that the CH node is chosen to be the origin (R_0) and we just want to

- **Lowest Aspect Ratio (LAR):** Select R_1 , and R_2 such that the triangle $\Delta(R_0, R_1, R_2)$ has the lowest aspect ratio among all different candidate triangles. This method usually selects nicely shaped triangles and avoids skinny triangles. However, it does not take into consideration the side lengths of the triangle.
- **Maximum Equilateral Triangle (MET):** This method is similar to LAR but takes the side length of the triangle into consideration. In this case, we search for all approximately equilateral triangles. Then we select the one with maximum side length. An approximately equilateral triangle is a triangle with aspect ratio close to 2. Hence, this method avoids skinny triangles while taking into consideration the side length of the triangle.
- **Highest Aspect Ratio (HAR):** Select R_1 , and R_2 such that $\Delta(R_0, R_1, R_2)$ has the highest aspect ratio among all different triangles. This method usually selects skinny triangles. We include this method to show how bad the accuracy could be if we are not careful selecting the coordinate system.
- **Minimum Initial Error (MIE):** In this method we try all possible local coordinate systems with origin R_0 located at the cluster-head node. For each coordinate system i , we calculate the initial position estimate P_0 using the MRLE algorithm, as described in section 5.8. Then we pick a coordinate system that gives an initial position estimate P_0 with minimum error function $E(P_0)$ given by Eq. 5.3. The intuition behind this method is to choose an initial position estimate P_0 such that the error function at this position $E(P_0)$ is as close as possible to 0. Although

select R_1 and R_2 .

this method is computationally expensive, we will show in the results section that it gives very acceptable accuracy that may lead to avoid the expensive refinement phase.

Table 5.1 compares between the above heuristics in terms of time complexity where d refers to the average node degree, n_c is the number of node per cluster, and e_c is the number of edges in the cluster. In section 5.10, we shall compare between the above methods in terms of the accuracy of the estimated node positions.

LCS Heuristic	Complexity
LAR	$O(d^2 n_c)$
HAR	$O(d^2 n_c)$
MET	$O(d e_c)$
MIE	$O(d^2 n_c^3)$

Table 5.1: Time complexity of different heuristics to select the local coordinate system (LCS)

5.7.2 Resolving Reflection

As discussed in section 5.6, a skinny triangle will exist if $\| \overrightarrow{u_2 R_2^u} - \overrightarrow{u_1 R_2^u} \| \leq \delta$. This means that we should select the base line to be as far as possible from the node u . Now for the selector resolver node (R_u^2), it should be selected as the neighbor node to u with the highest altitude from the base line among all other neighbors. Notice that R_u^2 must also belong to the set of identified nodes. So if we can not find a node R_u^2 such that

the reflection is resolved, we can postpone estimating the position of node u until more neighbors are identified.

5.8 The Multi-hop Relative Location Estimation (MRLE)

Algorithm

5.8.1 Definitions and Terminologies

- Identified Nodes Set, I , is the set of all nodes with known relative position with respect to the LCS. Initially, $I = \{R_0, R_1, R_2\}$. Similarly the set of unidentified nodes (U) is defined as $U = N_c - I$, where N_c is the set of all nodes inside the cluster.
- Identified Neighbors Set, $I(u) = I \cap N(u)$, is the set of neighbor nodes of u that have known relative position.
- Node Error Level, $EL(u)$, is the error level associated with node u as described in section 5.7.
- Measured distance between two nodes, $D(u, v)$, is the measured distance between nodes u and v as reported during the network bootstrapping phase.
- Candidate Positions for a node, u_1 and u_2 , are the two candidate positions for node u such that they are reflection of each other across the base line.

```

MRLE( $R_0, R_1, R_2$ )
1.  $I = \{R_0, R_1, R_2\}$ ;
2.  $U = N(R_0) \cap N(R_1) \cap N(R_2)$ ;
3. While there is no new identified nodes
4.   For each node  $u \in U$  and  $|I(u)| \geq 3$ 
5.      $[R_0^u, R_1^u] = \text{selectBaseLine}(u, I(u))$ ;
6.      $[u_x, u_1, u_2] = \text{findPositionUsingCosLaw}(u, R_0^u, R_1^u)$ ;
7.      $u_e = \text{selectCandidate}(R_0^u, R_1^u, u, I(u), u_1, u_2, \text{errorFlag})$ ;
8.     if  $\text{errorFlag} = \text{FALSE}$  // If reflection resolved
9.        $P_0(u) = u_e$ 
10.       $\text{EL}(u) = \text{mean}(\text{EL}(R_0^u), \text{EL}(R_1^u)) + 1$ ;
11.       $I = I \cup \{u\}$ ;
12.    end // for
13.  end // while
14. // Estimate position for nodes with circular dependency or with only two known ranges
15. For each node  $u \in U$  and  $|N(u) \cap I| \geq 2$ 
16.   Select  $R_0^u$  and  $R_1^u$  with the lowest error level and  $R_0^u \in I(u)$  and  $R_1^u \in I(u)$ ;
17.    $[u_1, u_2] = \text{findPositionUsingCosLaw}(u, R_0^u, R_1^u)$ ;
18.    $ec_1 = ec_2 = 0$ ;
19.   For each node  $v \in I - I(u)$ 
20.      $\Delta_1 = \| \overrightarrow{u_1 v} \|$ ;  $\Delta_2 = \| \overrightarrow{u_2 v} \|$ ;
21.     if  $\Delta_1 \leq T_r$ 
22.        $ec_1 = ec_1 + 1$ ;
23.     if  $\Delta_2 \leq T_r$ 
24.        $ec_2 = ec_2 + 1$ 
25.     if  $ec_1 < ec_2$ 
26.        $P_0(u) = u_1$ 
27.     else
28.        $P_0(u) = u_2$ ;
29.      $\text{EL}(u) = \text{mean}(\text{EL}(R_0^u), \text{EL}(R_1^u)) + 1$ ;
30.      $I = I \cup u$ ;
31.   end //for
 $[u_1, u_2] = \text{findPositionUsingCosLaw}(u, R_0^u, R_1^u)$ 
// Input:
//  $u \rightarrow$  The node to estimate its position.
//  $R_0^u, R_1^u \rightarrow$  The base line of  $u$ .
// return:
//  $u_1, u_2 \rightarrow$  the two candidate positions for node  $u$ .
32.  $d_1 = D(R_0^u, u)$ ,  $d_2 = D(R_1^u, u)$ 
33.  $d_3 = \| \frac{R_0^u R_1^u}{\| R_0^u R_1^u \|} \|$ ;
34.  $\vec{x} = \frac{R_0^u R_1^u}{\| R_0^u R_1^u \|}$ ;
35.  $\vec{y} = \frac{\vec{x}_\perp}{\| \vec{x}_\perp \|}$ ;
36.  $\cos(\theta) = (d_1^2 + d_3^2 - d_2^2) / (2d_1 d_3)$ ;
37.  $\delta_x = d_1 \cos(\theta)$ ;
38. // There are two candidates for the Y-coordinate
39.  $\delta_y = \sqrt{d_1^2 - \delta_x^2}$ ;
40.  $\vec{u}_1 = \vec{R}_0^u + \delta_x \vec{x} + \delta_y \vec{y}$ ;
41.  $\vec{u}_2 = \vec{R}_0^u + \delta_x \vec{x} - \delta_y \vec{y}$ ;

 $u_e = \text{selectCandidate}(R_0^u, R_1^u, u, u_1, u_2, \text{errorFlag})$ 
// Input:
//  $u \rightarrow$  The node to estimate its position.
//  $R_0^u, R_1^u \rightarrow$  The base line of  $u$ .
//  $u_1, u_2 \rightarrow$  Two different candidate positions for  $u$ .
// return:
//  $\text{errorFlag} \rightarrow$  true if we can not resolve reflection.
//  $u_e \rightarrow$  the estimated position of node  $u$  and  $\text{errorFlag} = \text{false}$  in this case.
42. Sort  $I(u)$  based on the altitude with respect to  $\overrightarrow{R_0 R_1}$ .
43. for  $R_2 \in I(u)$  starting from highest altitude first
44.    $l_1 = \| \vec{u}_1 - \vec{R}_2 \|$ ;  $l_2 = \| \vec{u}_2 - \vec{R}_2 \|$ ;
45.    $l = D(u, R_2)$ ;
46.    $\delta_1 = |l - l_1|$ ;  $\delta_2 = |l - l_2|$ ;
47.   if  $|(\delta_1 - \delta_2)| > \epsilon$ 
48.     if  $(\delta_1 < \delta_2)$ 
49.       return  $u_e = u_1$ ;
50.     else
51.       return  $u_e = u_2$ ;
52. end; // for
52.  $\text{errorFlag} = \text{true}$ ; // If we reached this point, this means we could not resolve reflection

```

5.8.2 The MRLE Algorithm

The MRLE algorithm starts by selecting the LCS reference nodes R_0, R_1 , and R_2 as described in section 5.7.1 and adds R_0, R_1 , and R_2 to the set of identified nodes (I). Then iteratively, try to estimate the position of an unidentified node (u) that has *three* or more distances to identified nodes using the technique described in section 5.3. If the node u has *two* or more distances to identified nodes, then MRLE uses the technique described in section 5.5 to estimate node position. The algorithm terminates when all nodes are identified or we have no more nodes with two or more distances to identified nodes. The algorithm details are given in Fig. 5.8.

5.9 The Refinement Step

The main goal behind the refinement step is to reduce the accumulated error in the initial position estimate using LSE optimization. This step is optional if higher accuracy is required. In the results section we will show that with careful selection of the initial LCS, we can achieve acceptable accuracy without the need for the expensive refinement step. The refinement step iteratively uses gradient descent method to refine the initial position estimates P_0 . The gradient $\nabla E(P)$, of the error function E (Eq. 5.3) can be calculated as follows:

$$\nabla E(P) = \begin{bmatrix} \frac{\partial E}{\partial x_0}(P) & \frac{\partial E}{\partial y_0}(P) & \frac{\partial E}{\partial z_0}(P) \\ \frac{\partial E}{\partial x_1}(P) & \frac{\partial E}{\partial y_1}(P) & \frac{\partial E}{\partial z_1}(P) \\ \vdots & \vdots & \vdots \\ \frac{\partial E}{\partial x_{n_c-1}}(P) & \frac{\partial E}{\partial y_{n_c-1}}(P) & \frac{\partial E}{\partial z_{n_c-1}}(P) \end{bmatrix} \quad (5.7)$$

The gradient ($\nabla E(P)$) is an $n_c \times 3$ matrix function of P , where n_c is the number of nodes in the cluster. It has the property that when it is evaluated at any position estimate P , it points in the direction of travel from P that will maximally increase the error (i.e., uphill). Therefore, to decrease the error (E), the value of P should be slightly changed in the opposite direction (i.e., $-\nabla E(P)$). The new value of P at iteration j is calculated as follows:

$$P_j = P_{j-1} - \lambda_j \nabla E(P_{j-1}) \quad (5.8)$$

Where P_0 is calculated using the MRLE algorithm. At each iteration, $E(P_j) < E(P_{j-1})$ as long as the parameter λ_j is small enough. The non-linear error function given by Eq. 5.3 has many local minima; so selecting the initial position estimate P_0 affects the accuracy of the estimated position significantly as well as the convergence latency. Since each different LCS will lead to a different initial position estimate (P_0), selecting the LCS will also affect the accuracy of the estimated position *after* the refinement step. This emphasize more on how important it is to select the best LCS. The terminating condition for the iterative minimization process is when the maximum change in any node position is $\leq \eta$, where η is the desired position accuracy.

5.10 Validation and Performance Evaluation

5.10.1 Experiments Setup and Goals

Both the MRLE algorithm and the refinement step were implemented using MATLAB 6.1 release 12.1. All experiments were performed over more than 1500 different cluster rep-

representing different cluster sizes (n_c) ranging from 20 to 60 nodes. For each topology, the transmission range of each node (T_r) was varied in order to achieve different node connectivity levels ranging from 7 to 17. The cluster radius (k) ranges from 2 to 4 depending on the cluster size and node connectivity. Initially, the nodes were randomly placed according to a uniform distribution on a 100x100 area. The inter-node distance measurements were perturbed with a Gaussian random noise with zero mean and variance σ^2 , where σ ranges from 0 to 8.

There are four parameters used in our simulation:

1. *Cluster size* (n_c): the number of nodes in the cluster including the cluster head node.
2. *Cluster radius* (k): the maximum number of hops between any node in the cluster and the cluster head node.
3. *Average Node Degree* (d): the average node degree in the cluster. Recall from section 4.5 that node degree is a function of the node transmission range (T_r).
4. *Range error* (σ): this is the measurement error associated with each distance between any two nodes. This is dependent on the technology used for distance estimation (TOA, AOA, RSSI). In the simulation, we assume that the TOA method is used; hence we assume Gaussian range error with zero mean and variance σ^2 .

We consider the following two performance metrics:

1. *Accuracy*: the accuracy of the estimated positions is measured in terms of the median error between the estimated positions and the true node positions.

2. *Convergence latency*: the number of iterations taken till the refinement step terminates (i.e. a minimum for the error function E (Eq. 5.3) is reached)

The overall goal of the following experiments is to quantify these metrics and qualify the impact of the various parameters. Mainly, we are interested in answering the following questions:

- *Q1*: Does selecting the local coordinate system (LCS) affect the accuracy of the estimated position and the convergence latency of the optimization? If so, how to select the local coordinate system? In the simulator, we are trying the four different heuristics described in section 5.7.1.
- *Q2*: What are the factors (cluster size, cluster radius, node degree, etc.) that affect accuracy, as the node becomes k -hops away from the cluster head node? Our goal here is to find different parameters that we can tune to obtain different levels of accuracy.
- *Q3*: If a good local coordinate system were selected, would the initial position estimates (P_0) be close enough to the positions resulting from the optimization? In other words, what added accuracy do we gain by conducting the optimization?

5.10.2 The Effect of Local Coordinate System (LCS) on Performance

The first set of experiments studies the effect of the selection of the local coordinate system on the accuracy of the estimated positions and how the network size impacts it. The effect of the selection of a local coordinate system on achieved accuracy is captured in

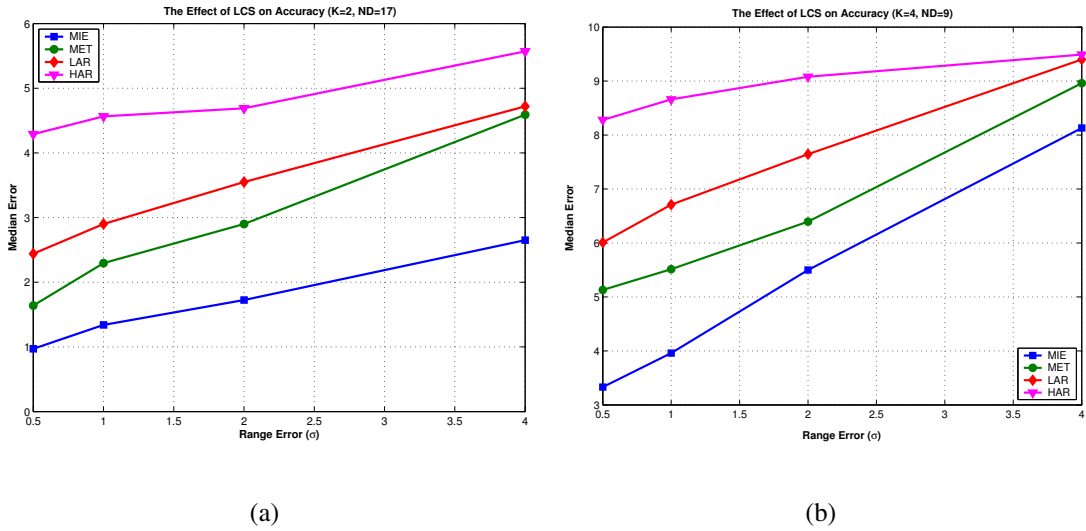


Figure 5.9: The effect of local coordinate system (LCS) on accuracy for different values of k and d

figure 5.9. In general, the experiments clearly indicate that selecting the local coordinate system is one of the most important factors affecting the accuracy of the estimated final nodes' positions. For high node degree ($d = 17$) and low cluster radius, the accuracy obtained if we use MIE is almost double the accuracy obtained using LAR or MET. The gap actually increases as the cluster size increases. The HAR curve shows how bad it could be if we do not carefully select the LCS. The figures also show that for low node degree, and as k increases, the error increases and the performance deteriorates regardless of which method we use for selecting the LCS.

From both figures, one can confirm that the MIE approach, which corresponds to minimum initial error $E(P_0)$, performs very well compared with other methods. The very good accuracy of MIE can be explained if we investigate the error function E (Eq. 5.3) closely. The function E is a function in $2n_c$ variables, assuming 2D coordinates, where n_c is the number of nodes in the clusters. The function has many local minima; hence; the

initial position estimate (P_0) is important since the refinement step can get stuck in one of the local minima. MIE is the only method that chooses an initial position estimate close to the optimal, and hence the probability of reaching a local minima decreases.

It is also interesting to note that the closeness to the performance of the LAR and MET methods, which is mainly due to the high similarity between the two methods. In the most part MET leads to slightly better accuracy because it considers triangles with large side length. Increasing the side length of the triangle reduces the effect of the error introduced by the range-estimation technology used (TOA). One more thing to notice from the figure is that the effect of range error is mostly symmetric on all methods of picking the local coordinate system, with an order of magnitude increase of error variance approximately worsening the accuracy by factor of 0.5. As the range error increases, the median error increases linearly.

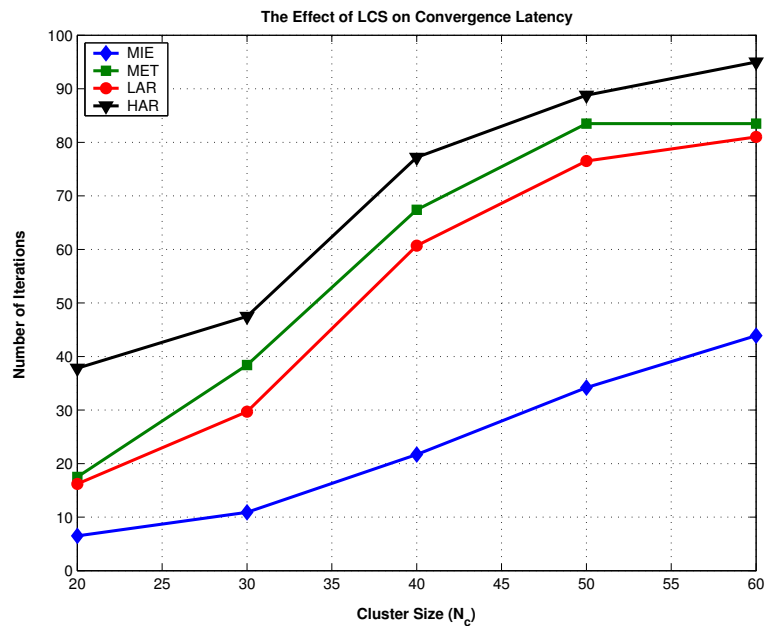
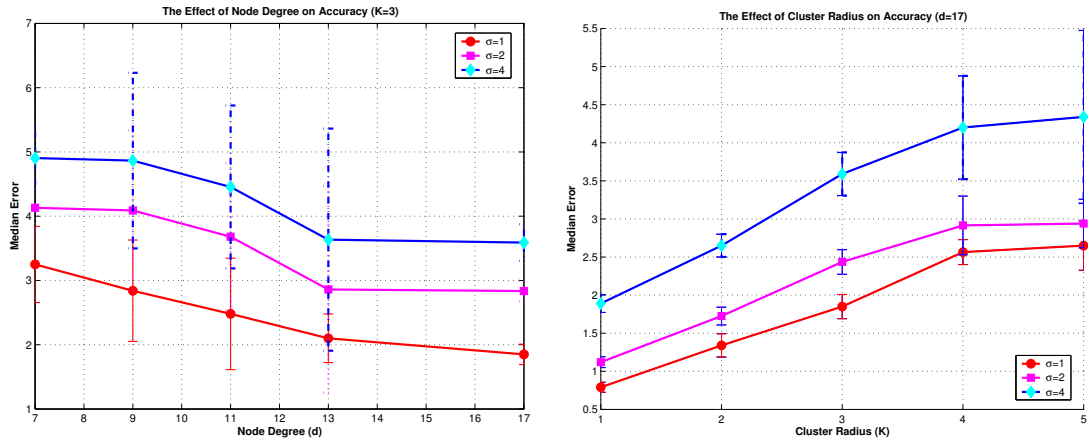


Figure 5.10: The effect of local coordinate system (LCS) on convergence latency

Fig. 5.10 reflects another implication of the local coordinate system, which is the convergence latency of the optimization. Not only has the MIE method performed well, as demonstrated by the figure, it actually expedite the convergence latency of the optimization. The convergence latency if MIE is used is almost half the latency if any other method is used. Although, MIS is computationally expensive compared with LAR and MET, we gain a lot during the refinement step. We can also see that the cluster size is the major factor affecting the complexity of the optimization. It is also worth noting that the number of iterations increases linearly with the growth in network size demonstrating the scalability of our approach. In the remainder of this chapter, if not explicitly mentioned, the MIE method will be used to select the local coordinate system (LCS).

5.10.3 Achievable Accuracy

In the second set of experiments, we report the achievable accuracy of our algorithm and captures the effect of cluster radius (k), node degree and range error. Figure 5.11 shows how the accuracy of the estimated position is affected by the node connectivity (d) and the cluster radius (k). The error bars represent 95% confidence interval. The effect of the range error is also captured in both charts. From Fig. 5.11(a), it can be concluded that increasing node degree has a very positive impact on the overall accuracy but it seems to saturate after a certain level ($d = 13$). It is also clear from the figures that increasing the connectivity decreases the uncertainty in the results. Fig. 5.11(b) shows that an increased value of the cluster radius worsens the accuracy and increases the uncertainty of the results. This is very much expected since the further the node is, the higher the

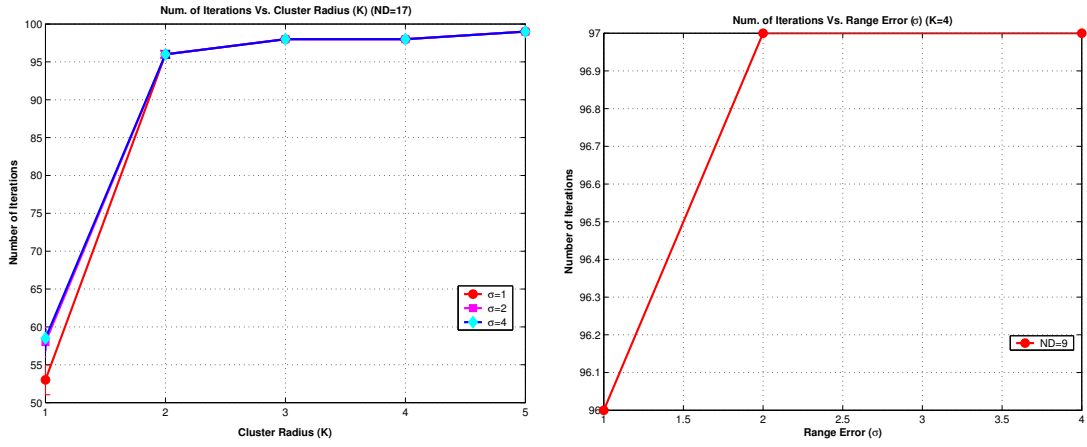


(a) The impact of average node degree (d) on accuracy (b) The impact of cluster radius (k) on accuracy

Figure 5.11: The effect of different simulation parameters on accuracy

accumulative range error becomes.

Combing the findings of figures 5.11(b) and 5.11(a), it can be concluded that as the cluster radius increases, the network connectivity should also be increases in order to maintain high accuracy. Looking into the problem in more details, we can see that the reason for this is that the number of constraints (edges) per node increase; hence; minimizes the possibility of reflection error. This means that we need to find a metric that related the number of edges in the local cluster graph (LCG) to the number of nodes (n_c). Notice also that the optimal case is when the local cluster graph is complete (i.e. each node is connected to all other nodes). In the OK clustering algorithm, section 4.5, we introduced the *CLIQUE Factor (CF)* as a measure of performance. The CLIQUE factor of a cluster measures how close the subgraph induced by cluster to a complete graph. Hence the CF related the number of edges in the LCG to the maximum number of edges we can have (optimal case). It turned out that the CF is the major parameter that affects



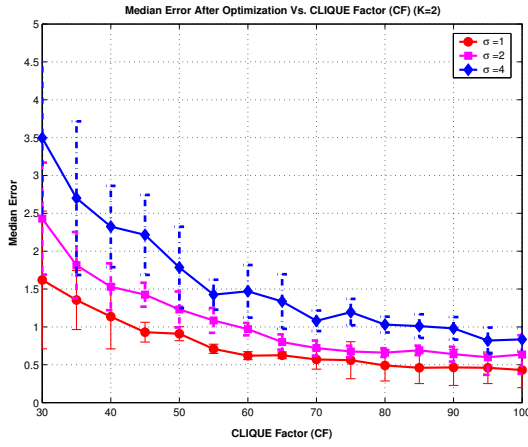
(a) The effect of average node degree (d) on convergence latency (b) The effect of cluster radius (K) on convergence latency

Figure 5.12: The effect of different simulation parameters on convergence latency

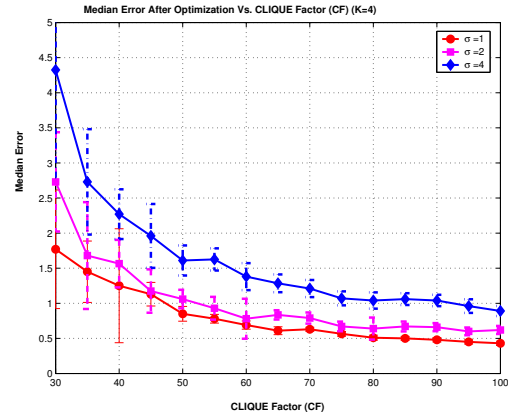
the accuracy within the cluster. The CF combine the effect of both cluster radius (k) and node degree (d) regardless of the cluster size.

Fig. 5.13 shows the impact of CLIQUE factor on accuracy for different values of k . We can see from the figures that the median error is high if the CF is less than 60%. The CLIQUE factor has slight effect on the accuracy beyond 60%. The 95% confidence-interval error bars is almost 0 as the CF increase. For acceptable accuracy, we recommend that the CF be at least 50%. The results shown in Fig. 5.13 explain the results show in Fig. 5.11. In Fig. 5.11(a), as the node degree increases within the cluster, the number of edges increase while the cluster size (n_c) is fixed. Hence, the CF increases and that's why the accuracy improves as node degree increases. When the cluster radius increases, Fig. 5.11(b), the cluster size increases while fixing the node degree (i.e. number of edges is fixed). Hence the CF decreases and the accuracy deteriorates.

One more interesting thing to notice is that the accuracy could be 60% of range-



(a) Cluster Radius ($k=2$)



(b) Cluster Radius ($k=4$)

Figure 5.13: The effect of CLIQUE factor (CF) on accuracy

error standard deviation if the CF is greater than 80%. The accuracy increases as the range error variance increases and can reach up to 75% as CF reaches 100% (i.e. complete graph). We think that this very interesting observation since this gives the sensor network engineer a trade-off between power and accuracy. Figure 5.14 shows the relation between the CLIQUE factor (CF) and node transmission range (T_r) for a square area of side length 100 distance units and node density 0.01, which is considered a low node density.

Finally, we want to study the impact of CLIQUE factor (CF) on convergence latency. As we may expect, Fig. 5.15 shows that the CLIQUE factor does not affect the convergence latency. Since changing the CF only changes the number of edges in the local cluster graph but not the cluster size, the convergence latency should not be affected.

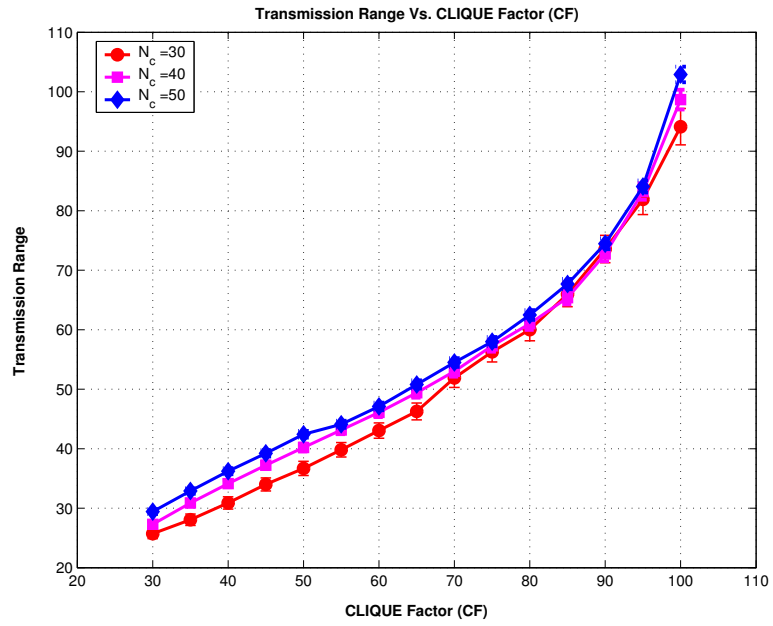


Figure 5.14: The relation between CLIQUE factor (CF) and node transmission range (T_r)

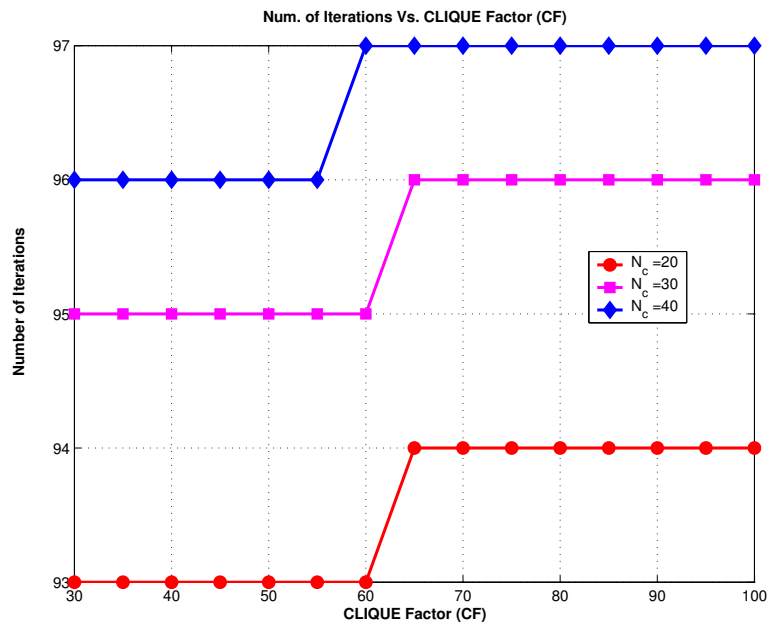


Figure 5.15: The effect of CLIQUE Factor (CF) on convergence latency

5.10.4 Optimization Factors

The last set is dedicated to the convergence latency and the added value of the optimization. We compare the quality of the optimized position estimate to that of the initial estimates under two different methods for picking the local coordinate system and qualify the value of conducting optimization. In Fig. 5.16, we try to show how much accuracy we gain by solving the non-linear optimization problem using the MIE method to select the local coordinate system (LCS). The figure compares between the accuracy of the estimated position before and after the refinement step for different range-error standard deviation (σ). It is clear from the figure that the refinement step increases the accuracy more than 100%. This gives the sensor network engineer another parameter to play with, the node computation power, since the non-linear optimization during the refinement step does require a lot of computation power. An interesting thing to notice from Fig. 5.16 is that the error in the estimated position using MIE is almost the same as the range error. The enhancement is much better as range error increases and the error in the estimated position can reach approximately 50% of the range error. This confirms the effectiveness of the MRLE algorithm and show how accurate the initial estimated position is. One thing to notice here is that the CLIQUE factor affects the accuracy of the estimated position whether optimization is used or not because increasing the CLIQUE factor reduces the probability of having reflection error which is the major source of localization error.

In Fig. 5.17, we compare between the accuracy of the initial position (i.e. before optimization) using just two different methods MIE and MET and the accuracy of the position obtained after performing optimization. The reason for selecting MET is that

it is not computationally expensive as MIE and gives acceptable accuracy in terms of initial position estimate. Hence, we consider MET as the method to use if the node has very low computational power capabilities. Clearly, the position estimated using MIE is more accurate than the position estimated using MET even after performing optimization. The figures gives the application a trade-off between computational power and accuracy. Table 5.2 summarizes different accuracy levels for range error $\sigma = 4$. Each row represents different CLIQUE factor (i.e. different transmission power) while each column represents different computational power.

CF	Low (MET NOPT)	Meium (MIE NOPT)	High (MIE OPT)
Low (40%)	4.5	3.25	2.25
Medium (60%)	3.75	2.75	1.5
High (100%)	2.75	2.25	1.25

Table 5.2: Trading accuracy with computational power and transmission power

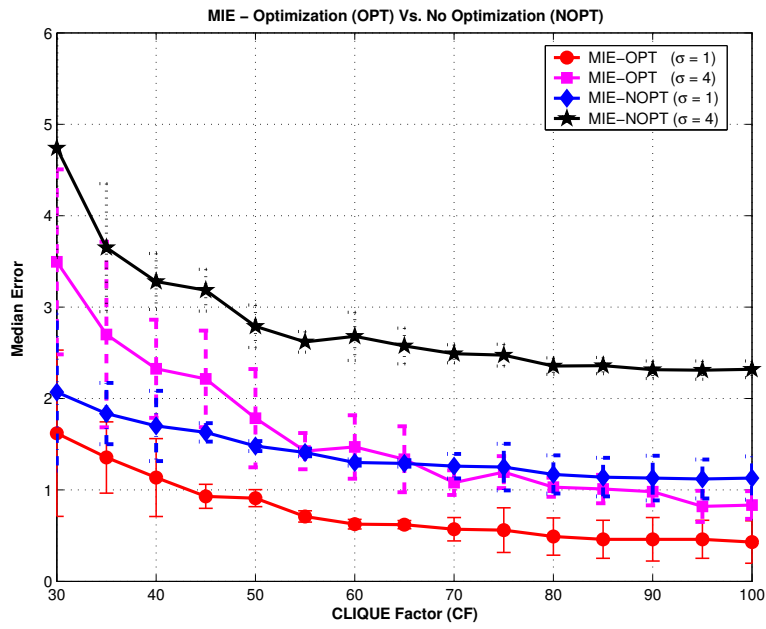
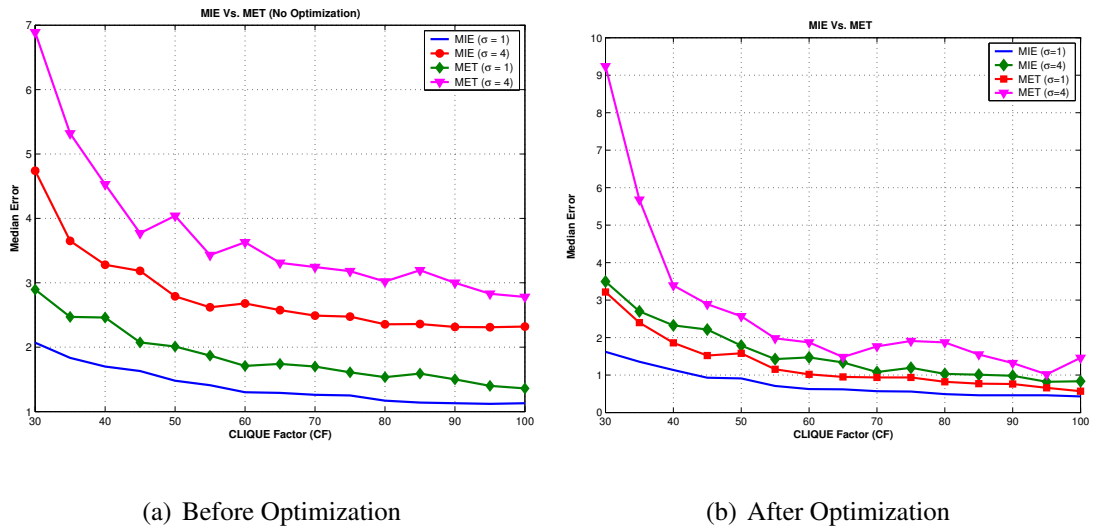


Figure 5.16: The accuracy before and after optimization using MIE to select LCS



(a) Before Optimization

(b) After Optimization

Figure 5.17: A comparison between accuracy before and after optimization using MIE and MET for selecting the LCS

Chapter 6

The Global Location Discovery Phase

In the *GLD* phase, the cluster head nodes collaborate to obtain a global map of the network. After forming a cluster-level map during the *LLD* phase, the local maps have different directions. Two local maps have the same direction if their x -axes are pointing in the same direction and similarly for the y -axes (and z -axes in case of 3-D). A global coordinate system can be built from the local maps available at each cluster head using simple matrix rotations, translations, and mirroring. In this chapter, we describe how to adjust the directions of the local maps of the cluster head nodes to obtain the global topology of the network using *boundary nodes* and show how the number of boundary nodes (overlapping degree) between two clusters affect the accuracy of the transformation from one coordinate system to another.

As described in the SALAM system model, section 3.2, we assume that the sensor nodes are capable of long-haul communication. Hence all cluster head nodes are capable of communicating directly with each other. Using this assumption, we will discuss in more details how to construct the *overlapping graph*¹ of the cluster heads and propose

¹The definition of the overlapping graph is given in section 4.3.1.

four different heuristics to assign weights to the edges of the overlapping graph. We also introduce a new problem, the *best order of transformations*, and show that it maps to finding a spanning tree for the overlapping graph. In the results section, we show how the spanning tree affect both accuracy and communication overhead of the GLD phase. One last thing to notice is that the GLD phase can be optional if a global view of the network is not needed, e.g. when the cluster head nodes do not perform joint application tasks. It is up to the application layer to decide whether to perform GLD phase or not.

6.1 The Best Transformation Matrix Problem

In order to compute the transformation matrix between two clusters, there must be at least three *boundary nodes* that belong to both clusters (i.e. within k -hops from both cluster head nodes). Since range measurements are typically inaccurate, we do not expect to find a transformation that maps the node coordinates of one cluster exactly into the measured coordinates of these nodes in the other cluster. Instead we need to formulate and solve another optimization problem by minimizing the sum of the squares of the residual errors as follows:

Let C_1 and C_2 be two adjacent clusters that have m common boundary nodes and $m \geq 3$. Let $v_i(C_1)$ and $v_i(C_2)$ be the coordinates of boundary node i in C_1 and C_2 respectively, where $i = 1, \dots, m$. We will refer to C_1 as the *child cluster* and to C_2 as the *parent cluster*. The objective is to find the transformation matrix M_{CP} that maps node coordinates of C_1 (*child cluster*) into C_2 (*parent cluster*), and minimizes the following error function:

$$\min E(M_{CP}) = \sum_{i=1}^m \|v_i(C_2) - M_{CP}v_i(C_1)\|^2 \quad (6.1)$$

$$\text{where } M_{CP} = \begin{bmatrix} r_1 & r_2 & t_x \\ r_3 & r_4 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

where t_x, t_y are the translation transformation, r_i is the rotation transformation and

possibly mirroring with the following properties:

$$\begin{cases} |r_1| = |r_4| \\ |r_2| = |r_3| \\ r_1r_4 - r_2r_3 = -1 \quad \text{formirroring, } 1 \quad \text{otherwise} \end{cases} \quad (6.2)$$

There is a closed form solution to the above minimization problem as described in [53] that takes $O(m)$. Apparently, increasing the overlapping degree (m) between the two clusters will reduce the error due to transformation. In the results section, we will analyze the effect of the overlapping degree on the accuracy of the estimated position.

6.2 The Overlapping Graph

The overlapping graph is defined in section 4.3.1 as follows:

Let S be the set of cluster head nodes. Then the *Overlapping Graph* (OG), G_S , is the weighted graph induced by S as follows:

1. The set of vertices are S .
2. An edge exists between two vertices u, v iff $N_k[u] \cap N_k[v] \geq \omega$, where N_k is the

closed neighbor set, and ω is some threshold representing the minimal overlapping degree between any two clusters.

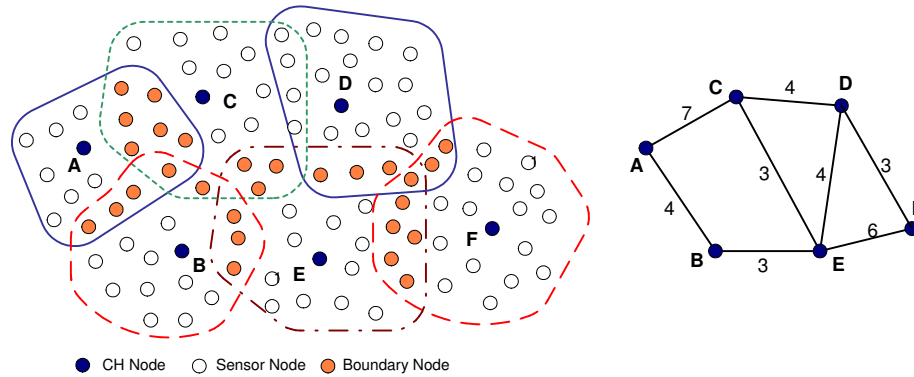


Figure 6.1: The overlapping graph

Each vertex in the overlapping graph represents one cluster in the network where an edge between two vertices implies that there is some overlapping between the two clusters. The threshold ω is application dependent. In SALAM, in order to be able to transform one local coordinate system to another, we need at least three boundary nodes (i.e. a minimum overlapping degree of 3); hence; we set $\omega = 3$). Therefore if an edge exists between two vertices u and v , this means that we can transform from the local coordinate system corresponding to u to that of v and vice versa.

Fig. 6.1 shows an example of the overlapping graph. There are six clusters in the network; hence; $S = \{A, B, C, D, E, F\}$. Each vertex in the graph represents one cluster. An edge exists between two vertices if the corresponding clusters overlap with more than 3 (i.e. $\omega = 3$). The edge weights represent the overlapping degree between the two corresponding clusters. Notice that there is no edge between B and C although the

corresponding clusters are adjacent because the overlapping degree is 2 (< 3).

The overlapping graph is a weighted graph. The edge weights are also application dependent and can be calculated according to different design goals. In the next section, we propose different heuristics to calculate the edge weights to optimize a certain design objective. Finally, the overlapping graph could be undirected or directed graph based on how the weights are calculated. The graph shown in Fig. 6.1 is undirected since the weights correspond to the overlapping degree between clusters.

6.3 The Best Order of Transformations Problem

In order to build a global network topology, we need to transform from one coordinate system to another as described in section 6.1. Given an overlapping graph of m cluster head nodes, we need to perform $m - 1$ transformations in order to merge m local coordinate systems into one big global coordinate system (GCS). This is equivalent to finding a spanning tree² (ST) for the overlapping graph. Given an overlapping graph, there are many spanning trees that links the cluster head nodes together. Each spanning tree corresponds to a different order of transformations between the local coordinate systems. Each order of transformations will result in a different accuracy, and different communication overhead per node. We will refer to the problem of finding the spanning tree, that satisfies a certain design goal, as *the best order of transformations* problem. In SALAM we are

²A spanning tree is a connected, acyclic subgraph containing all the vertices of a graph. Informally, a spanning tree of a graph is a selection of edges from the graph that form a tree spanning every vertex; that is, no vertex is not connected to the tree.

interested in finding the best order of transformations (*spanning tree*) that satisfy one of the following design goals:

1. Reduce average communication overhead per node to build a global map.
2. Reduce inter-cluster accumulated error; hence; minimize the overall global node position error and enhance the accuracy of the estimated position.

Finding a spanning tree is dependent on how the edge weights are interpreted. We propose the following heuristics to assign weights to the edges of the overlapping graph in order to satisfy one of the above design goals:

1. *Euclidian Distance (ED)*. Setting the weight of the edge between two cluster head (CH) nodes as the Euclidian distance between the two node aims at minimizing the communication overhead per CH node. In this case the overlapping graph is undirected graph. In order to estimate the Euclidian distance between the two CH nodes, we have two cases. If the two CH nodes are within k hops from each other, then the Euclidian distance can be calculated based on the estimated position during the LLD phase. Otherwise, the distance between the two CH nodes could be estimated as the number of hops between the two CH nodes multiplied by the transmission range (T_r) or the average edge length calculated from the LCG.
2. *Weighted Euclidian Distance (WED)*. This method also aims at reducing the communication overhead per node. It is similar to the previous heuristic except that the Euclidean distance between the two nodes is weighted by the number of non-boundary nodes between the two clusters. The overlapping graph is also undirected graph in this case.

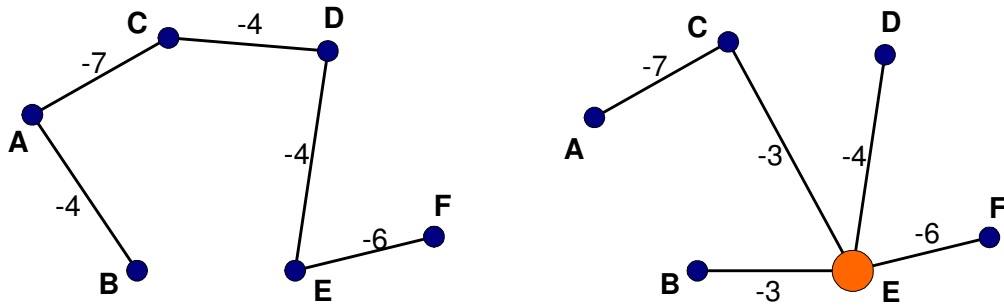
3. *The CLIQUE Factor (CF)*. In section 5.10.3, we have shown the effect of CLIQUE factor on accuracy. Given two cluster head nodes u and v with corresponding CLIQUE factors u_{CF} and v_{CF} ; respectively; where $u_{CF} > v_{CF}$, it is better to transform from the local coordinate system of v to that of u in order to enhance the accuracy of the estimated position. Hence, the weight of the edge connecting v to u ($w(v, u)$) is set to be u_{CF} and $w(u, v) = v_{CF}$. Clearly, the overlapping graph is a directed graph in this case.

4. *LLD Error (LE)*. In section 5.7.1, we have shown that selecting the local coordinate system with minimum initial error (MIE) leads to a highly accurate position estimate. This means if we have two cluster head nodes u and v where the error value of the estimated position, as calculated by Eq. 5.3, is E_u and E_v , where $E_u < E_v$, it is better to transform from the local coordinate system of v to that of u in order to enhance the accuracy of the estimated position. Hence, the weight of the edge connecting v to u ($w(v, u)$) is set to be E_u and $w(u, v) = E_v$. The overlapping graph is also a directed graph in this case.

In the result section, we compare between the above heuristics in terms of communication overhead and achievable accuracy. In the next section, we shall discuss two different approaches to find a spanning tree for the overlapping graph.

6.4 The Spanning Tree of The Overlapping graph

Given a weighted overlapping graph, we propose two approaches to find the spanning tree. The two approaches result in a spanning trees (ST) of different heights as follows:



(a) Minimum spanning tree

(b) Minimum height spanning tree

Figure 6.2: Different methods for finding the spanning tree of the overlapping graph

1. ***Finding the Minimum Spanning Tree (MST)***. The minimum spanning tree is the minimum-weight tree in a weighted graph which contains all of the graph’s vertices. There are two well-known algorithms to solve the MST problem: Prim’s algorithm, and Kruskal’s algorithm. In the current implementation, we use Kruskal’s algorithm. Fig. 6.2(a) shows the MST of the overlapping graph given in Fig. 6.1.

2. ***Finding the Minimum Height Spanning Tree (MHST)***. The distributed implementation for the minimum height spanning tree is based on Bellman Ford algorithm [43, 58]. The specific algorithm is available at [34]. To build a minimum height spanning tree, we use the minimum height spanning tree heuristic described in [60] as follows:

- Using Floyd-Warshall algorithm, we compute the length of the shortest paths between all pairs of nodes.

- Assign a weight for each node equal to the maximum shortest path length emanating from that node.

- Select the node with the smallest weight to be the root of the base tree.
- Create the spanning tree by merging the shortest paths from root to all other nodes in the overlapping graph.

Fig. 6.2(b) shows the MST of the overlapping graph given in Fig. 6.1.

In the result section, we compare between the two approaches and show how the height of the spanning tree affects both the communication overhead and the accuracy of the estimated global node position.

6.5 The GLD Algorithm

The GLD phase runs in parallel on all nodes. We assume that all CH nodes communicate with each other and exchange information that can be used to assign weights to the edges of the overlapping graph. For example, if we are going to use the CF heuristic, the CH nodes exchange the CF of their corresponding clusters. After that each CH node build a spanning tree (MST or MHST) by executing either Kruskal's MST or the MHST algorithm described in the previous section. Now each CH node has a spanning tree of the overlapping graph. The root of the spanning tree will be the origin of the global coordinate system (GCS). Hence, we start transforming from one coordinate system to another starting from the leaf nodes. Leaf nodes send the coordinates of non-boundary nodes to their corresponding parent clusters in the spanning tree. Then, The parent cluster merge all the child clusters with its own cluster and remove the leaf nodes (children) from the spanning tree; hence, the parent cluster becomes a leaf node. The process is repeated until

there is only one node in the tree. The algorithm is highlighted as follows:

1. After terminating the LLD phase, all cluster head nodes exchange information using long-haul communication in order to build a local copy of the overlapping graph at each CH node. The information exchanged contains data that can be used to assign weights to the edges of the overlapping graph.
2. Using the above information, each CH node constructs the overlapping graph. An edge exists between two CH nodes if the corresponding clusters have three or more boundary nodes. Each edge is assigned a weight based on one of the heuristics as discussed in the previous section.
3. Each CH node constructs either the minimum spanning tree (MST) or the minimum height spanning tree (MHST) for the overlapping graph.
4. If the CH node is a leaf, it transmits its local map to the parent in the MST and just wait to receive the global map before terminating the GLD phase.
5. Each non-leaf parent node receives the local maps of its children and removes the children from the MST; hence the parent becomes a leaf node. The parent finally calculates the joint cluster map by transforming all children clusters to its own coordinate system.
6. Repeat steps 3 and 4 until there is only one edge in the MST as shown in Fig. 6.3.
7. Now we have only two CH nodes left in the graph as shown in Fig. 6.3. In order to determine which node should be the root of the MST, we use the following rules:

- Select the CH node which has more joint clusters as the root of the tree because it has more children nearby and most probably is near the center of the network.
 - If the two CH nodes has the same number of joint clusters, select the one with lower node ID as the root.
8. Finally, the root of the MST broadcasts the global map of the network to the CH nodes either by direct communication or by multi-hop using the children in the MST.

Fig. 6.3 shows a step-by-step example of applying the GLD algorithm on the overlapping graph shown in Fig. 6.1. We assume that the MHST approach is used to construct a spanning tree of the graph.

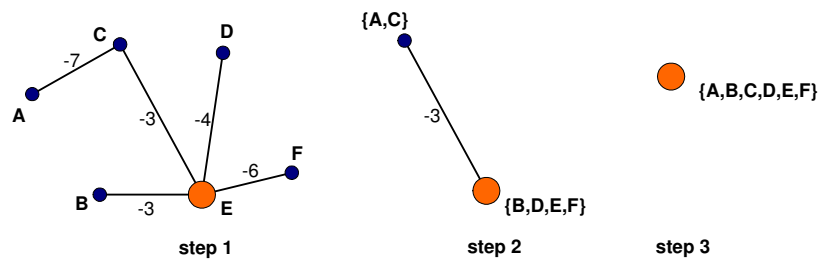


Figure 6.3: An example of GLD algorithm

6.6 Validation and Performance Evaluation

6.6.1 Experiments Setup and Goals

The GLD algorithm was implemented using MATLAB 6.1 release 12.1. All experiments were performed over more than 150 different topologies representing representing different network sizes (n) ranging from 50 to 800 sensor nodes. The nodes were randomly placed according to a uniform distribution on a 100x100 area. For each topology, the transmission range of each node (T_r) was varied in order to achieve different node connectivity levels (d) ranging from 7 to 17. The cluster radius (k) ranges from 1 to 5 depending on the cluster size and node connectivity. This configuration leads to an average CLIQUE factor (CF) ranging from 10% to 40%. The inter-node distance measurements were perturbed with a Gaussian random noise with zero mean and variance σ^2 , where σ ranges from 0 to 8.

There are five parameters used in the GLD simulation experiments:

1. *Number of Clusters (m)*: this is the number of clusters in the network. This corresponds to the number of nodes in the overlapping graph.
2. *Spanning Tree Weight (W)*: this corresponds to how the weights of the overlapping graph are calculated. In the simulation, we tried the four different heuristics: ED, WED, CF, and LE as described in section 6.3.
3. *Spanning Tree Height (STH)*: this corresponds to how the spanning tree is constructed. In the simulation, we tried two methods: minimum spanning tree (MST) and minimum height spanning tree (MHST), as described in section 6.4.

4. *The Average CLIQUE Factor (CF)*: the average CLIQUE factor of the network taken overall clusters.
5. *The Average Overlapping Degree (AOD)*: AOD is defined as the average overlapping degree between any two overlapping clusters in the network. Assume that u, v are any two cluster head (CH) nodes. Then the overlapping degree between the two corresponding clusters (\mathbf{O}) is a discrete random variable where $\mathbf{O} = |N_k[u] \cap N_k[v]|$ and $N_k[u] \cap N_k[v] \neq \emptyset$. Notice that the overlapping degree is defined only for overlapping clusters (i.e. the random variable \mathbf{O} can not take the value 0). We define *AOD* as the mean of this random variable \mathbf{O} (i.e. $AOD = E(\mathbf{O})$).
6. *Range error (σ)*: this is the measurement error associated with each distance between any two nodes. Like the LLD phase, we assume that the TOA method is used; hence we assume Gaussian range error with zero mean and variance σ^2 .

We consider the following two performance metrics:

1. *Accuracy*: the accuracy of the global estimated positions is measured in terms of the median error between the estimated positions and the true node positions.
2. *Communication Overhead*: this metric measures the average energy spent in communication per cluster head (CH) node. We assume a simple model for the radio hardware energy dissipation where the transmitter dissipates energy to run the radio electronics and the power amplifier. For the experiments described here, we use the free space channel model [49]. Thus, to transmit an b -bit message a distance l , the

radio expends

$$\begin{aligned} E_T(b, l) &= E_{T-elec}(b) + E_{T-amp}(b, l) \\ &= bE_{elec} + b\epsilon_{fs}l^2 \end{aligned} \tag{6.3}$$

The electronics energy, E_{elec} , depends on factors such as the digital coding, modulation, filtering, and spreading of the signal, whereas the amplifier energy, $\epsilon_{fs}l^2$, depends on the distance to the receiver. For the experiments described in this section, the communication energy parameters are set as: $E_{elec} = 50nJbit$, and $\epsilon_{fs} = 10pJ/bit/m^2$.

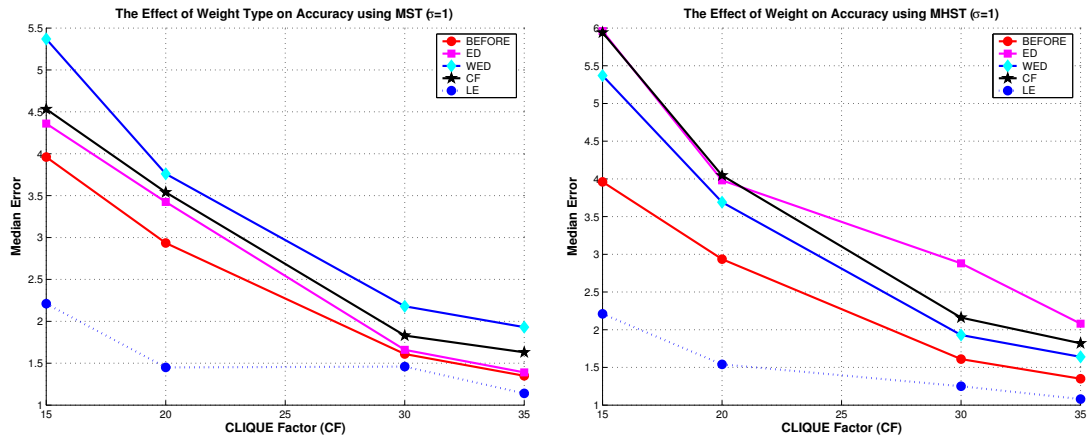
The overall goal of the following experiments is analyze the following:

- *Q1*: The effect of the overlapping graph spanning tree on accuracy. Two things to consider here, how the weights of the graph are calculated and the effect of the spanning tree height on accuracy. Again, our goal here is to find different parameters that we can tune to obtain different levels of accuracy.
- *Q2*: The effect of spanning tree height and weight heuristic on communication overhead per cluster head node. The objective is to give the sensor network engineer different parameters that can be tuned to obtain different levels power consumption and to confirm that SALAM is scalable during the GLD phase.
- *Q3*: The effect of GLD phase on the accuracy of the global estimated position and how much error the GLD phase introduces to the overall position estimation process. We will also study the impact of the overlapping degree between two clusters on the accuracy of.

6.6.2 The Effect of Spanning Tree on Accuracy

In the first set of experiments we report the accuracy of GLD phase and capture the effect of the overlapping graph spanning tree (ST) on the accuracy of the global estimated positions. The effect of the spanning tree weight on achieved accuracy is captured in figure 6.4. The figures compare between the accuracy of the estimated position before GLD phase (BEFORE) and after GLD phase using different heuristics to calculate edge weights. Initially, one would think that after performing the GLD phase, the error in the estimated position will increase due to transformation error. However, this is not always true. As shown from the figure, the experiments clearly indicate that setting the weight of the overlapping graph edges using the LE method leads to *improving accuracy* regardless of how the spanning tree is constructed (MST or MHST). The accuracy is almost doubled in case of low CF (10-20%). However, as the CLIQUE factor (CF) increases, the improvement decreases. This gives the sensor network engineer an inexpensive method to enhance accuracy for low CF (i.e. low transmission power). We will see in the next section that the effect of LE heuristic on communication overhead per node is not that bad. Using other heuristics to calculate edge weights may result to reducing communication overhead per CH node but may lead to decreasing the accuracy of the global estimated position up to 50%.

Fig. 6.5 reflects the implication of the effect of spanning tree height on accuracy using LE and ED methods to calculate weights. Using the LE method with minimum height spanning tree (MHST) leads to slight improvement in the accuracy as compared with using the MST. This can be explained as follows. As the height of the spanning



(a) using MST

(b) using MHST

Figure 6.4: The effect of spanning tree weight on accuracy

tree increases, the number of coordinate system transformations also increases, leading to an increase in the inter-cluster transformation error. Using a minimum height spanning tree will result in minimizing the number of transformations, which in turn leads to better accuracy. However, this does not hold in case of ED method since the primary goal there is to minimize communication overhead to enhance accuracy.

Finally, we can clearly see from the figures that the CLIQUE factor (CF) is still on of the major factors affecting accuracy. Increasing the average CF of the network leads to a more accurate estimated global nodes' positions and mutes the inaccuracy added by the GLD phase.

6.6.3 The Effect of Spanning Tree on Communication Overhead

The second set of experiments analyzes the communication overhead per node during the GLD phase and studies the effect of the spanning tree (ST) on communication overhead. Fig. 6.6 shows the impact of spanning tree weight on communication overhead per CH

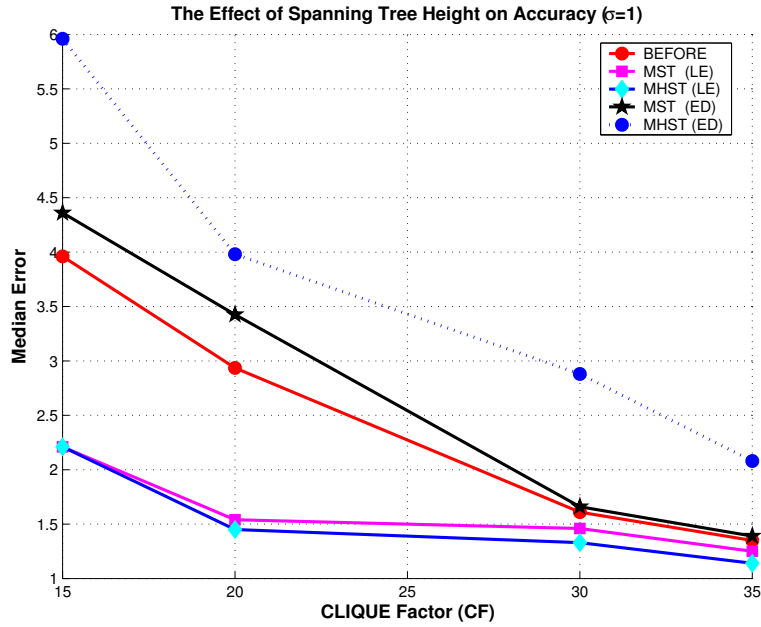
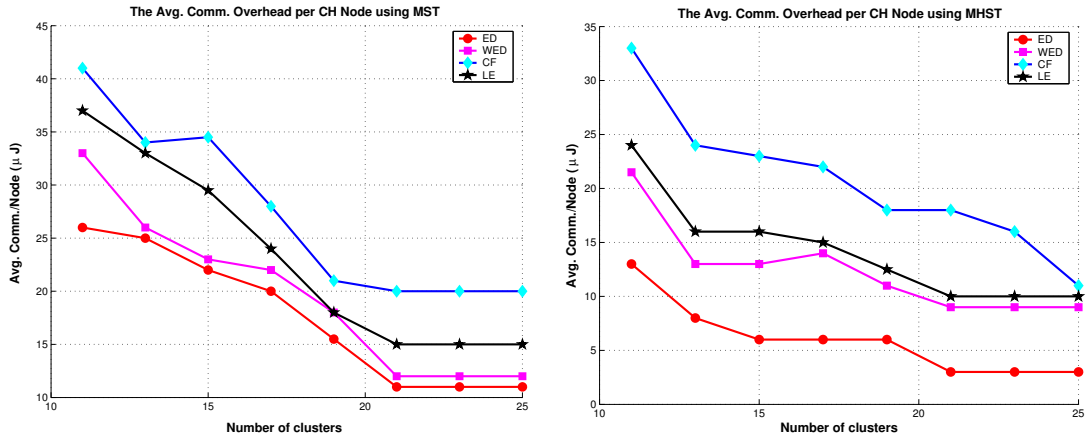


Figure 6.5: The effect of spanning tree height on accuracy

node as the number of clusters increase. We observe that the ED heuristic leads to the minimum communication overhead per node regardless of the tree height. We also notice that although LE method improves the accuracy, it costs almost double the communication overhead per node as compared with ED specially in the case of MHST. It is clear from the figures that the communication overhead per node is constant as the number of clusters increase. This, with the results discussed in section 4.5.3, confirms that SALAM is scalable in terms of communication overhead.

In Fig. 6.7 we study the effect of spanning tree height on communication overhead per CH node using the two different heuristics aiming at minimizing communication overhead (ED and WED). We observe that the minimum highest spanning tree (MHST) always leads to minimum communication overhead per node regardless of how the weights are calculated (i.e. ED or WED). We can also see from the figure that the energy con-



(a) Using MST

(b) Using MHST

Figure 6.6: The effect of spanning tree weight on communication overhead per node

sumed per node slowly decreases as the number of cluster increases from 10 to 20. Then it remains almost constant afterwards.

6.6.4 Achievable Accuracy

In the last set of experiments, we report the achievable accuracy of SALAM and compare between the accuracy before and after performing the GLD phase for different values of range error. We shall use the MHST method to construct the spanning tree since it leads to better accuracy as compared with MST. In order to simplify the graphs, we shall compare between only two heuristics to calculate edge weights: ED which leads to minimum communication overhead per node and LE which leads to minimum error.

Fig. 6.8(a) reports the improvement in accuracy using the LE method after performing GLD. We can see that we have 100% improvement in accuracy for low CLIQUE factor regardless of the range error. However, from Fig. 6.8(b), we can notice that if the ED method is used, the GLD phase will slightly decrease the accuracy. The error resulting

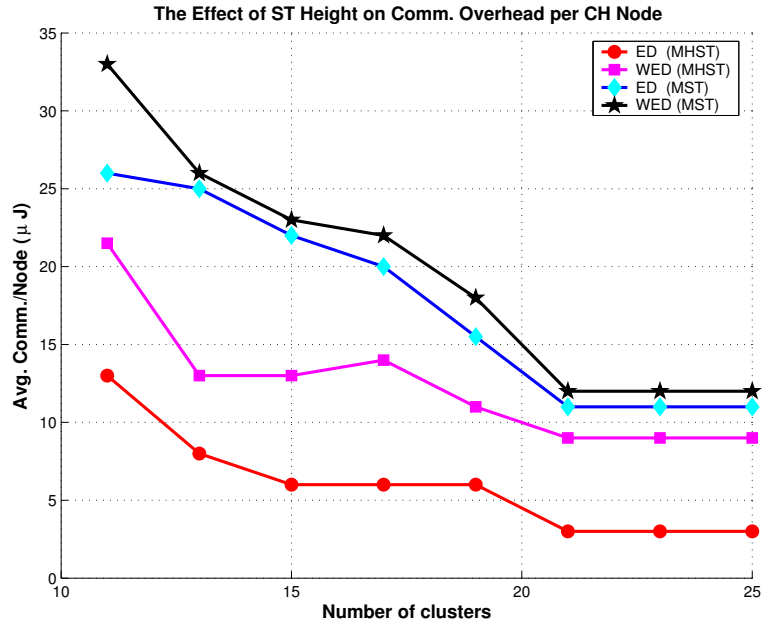


Figure 6.7: The effect of spanning tree height on communication overhead per node from the GLD phase is minimal compared to the error introduced during the LLD phase (e.g. reflection error). Increasing the CLIQUE factor always guarantees better accuracy.

Finally, we want to show the effect of average overlapping degree (AOD) between clusters on the accuracy of the estimated nodes' positions. From Fig. 6.9 we can see that increasing the AOD decreases the error introduced by the GLD phase because of the inaccurate transformation matrix (Eq. 6.1). Clearly, if the AOD is increases beyond 20, the effect is minimal and almost no change in accuracy for $AOD > 30$.

6.7 Comparison With Other Localization Techniques

In this section, we compare the accuracy of SALAM with the accuracy of different ad-hoc localization algorithms. We chose a subset of algorithms that represents a variety of different taxonomy features, as discussed in chapter 2. The following algorithms will be

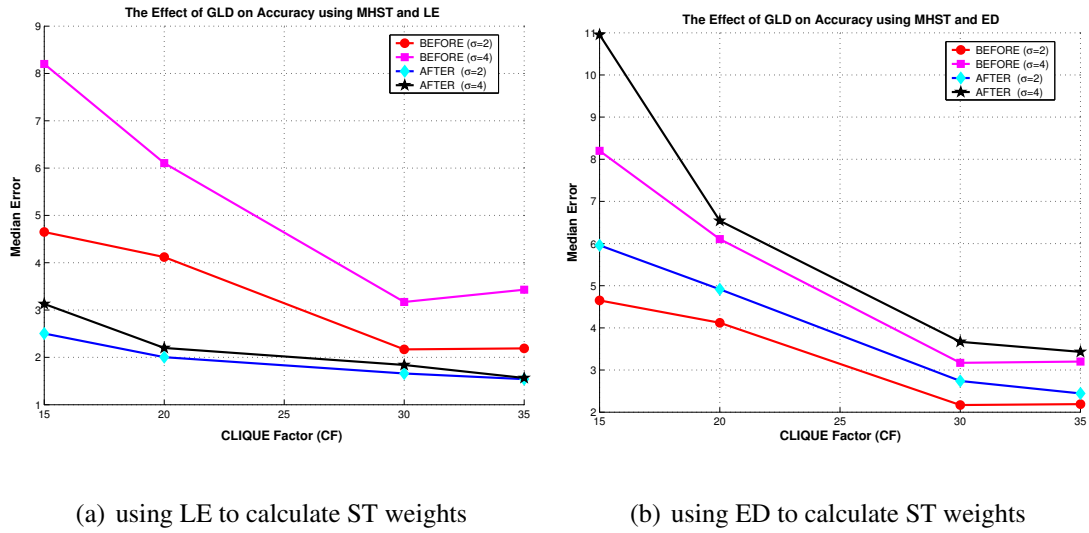


Figure 6.8: The effect of GLD phase on accuracy

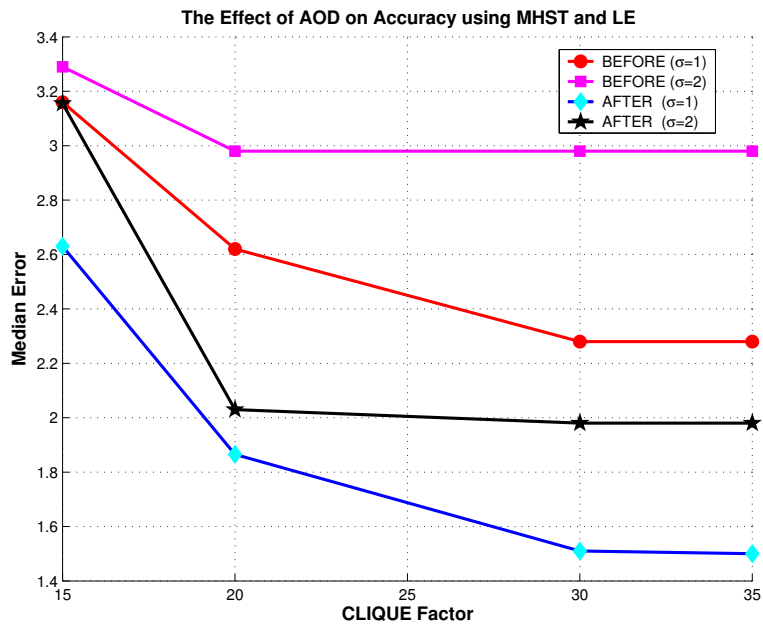


Figure 6.9: The effect of average overlapping degree (AOD) on accuracy

used in the comparison:

1. **MDS-MAP** algorithm [80] is a localization method based on multidimensional scaling (MDS). It is classified as a centralized anchor-free range-free algorithm.
2. **MDS-MAP(P)** algorithm [79] is an improved version of MDS-MAP. Like SALAM, MDS-MAP(P) is also cluster-based, where $k = 2$. It is classified as a distributed anchor-free range-based algorithm. We consider MDS-MAP(P) as the highest competitor localization algorithm to SALAM. MDS-MAP(P) can use an optional refinement phase to enhance the position accuracy using least squares minimization. In this case, we shall refer to the algorithm as MDS-MAP(P,R).
3. **Convex position estimation** algorithm [37] is a well-known centralized localization algorithm. It is classified as a centralized anchor-based range-free algorithm.
4. **Hop-TERRAIN** algorithm [74] represents the class of distributed anchor-based range-based algorithms.

Fig. 6.10 compares between the accuracy of SALAM and MDS-based algorithms on networks with 200 nodes uniformly distributed in a square field with side length = 100. The x -axis represents connectivity (i.e. average node degree) and the y -axis represents the median error as percentage of transmission range (T_r). The node transmission range ranges from 12.5 to 25, with an increment 2.5, which lead to average connectivity levels 8.8, 12.3, 16.4, 20.9, 25.9, and 31.1, respectively. The corresponding average CLIQUE factor ranges from 20 to 50. The range error standard deviation (σ) is equal to 0.5.

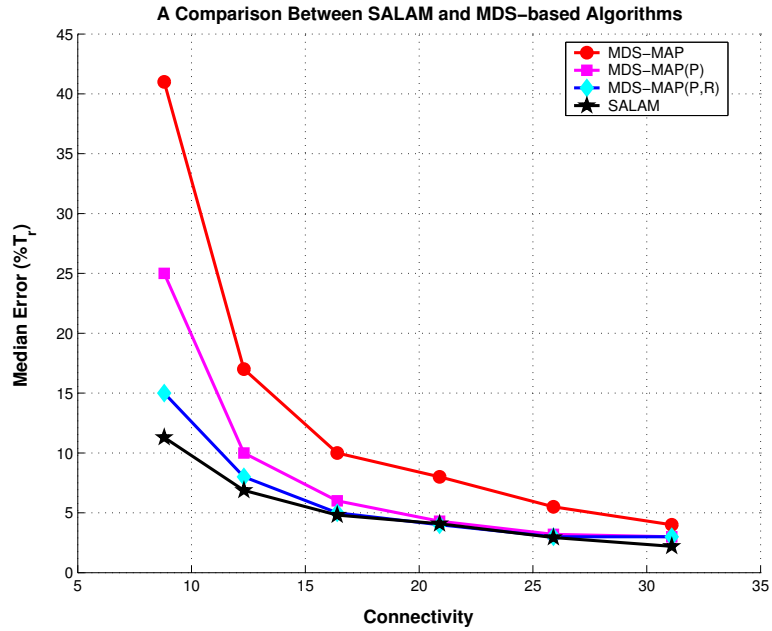


Figure 6.10: A comparison between SALAM and MDS-based algorithms using uniform topology

Although, MDS-MAP algorithm is a range-free algorithm, the results shown in Fig. 6.10 are based on ranges not connectivity information. The MDS-based techniques use *ten* anchor nodes to achieve the accuracy shown in the figure. However, SALAM uses only *three* anchor nodes. The curves show that SALAM is consistently better than the basic MDS-MAP technique and is more than $30\%T_r$ better when the connectivity is low. Compared with the improved version (MDS-MAP(P)), SALAM is approximately $15\%T_r$ better for low connectivity ($j \leq 16$). For higher connectivity, the accuracy is almost the same. However, the authors reported in [79] that as connectivity increases, the accuracy does not improve. This is actually show in the figure as connectivity goes beyond 25, the error is almost the same. This is not the case in SALAM. In Fig 5.13, we have shown that increasing the connectivity (i.e. implicitly increasing the CLIQUE factor) the

accuracy increase. The reason for this is that SALAM uses the new added edges to resolve reflection and during the refinement phase. We can also see from the figure that even after performing the refinement step (MDS-MAP(P,R)), SALAM is still approximately $5\%T_r$ better than MDS-MAP(P,R) for low connectivity. Please keep in your mind that the above results of MDS-based techniques is when using 10 anchor nodes while SALAM is using just three nodes.

In [79], the authors also done experiments on grid networks. Figure 6.11 compares the results of SALAM, using uniform topology, and MDS-based, using grid networks with 4 random anchors. Although MDS-based techniques obtain much better results than on the random networks, they could not outperform the accuracy of SALAM.

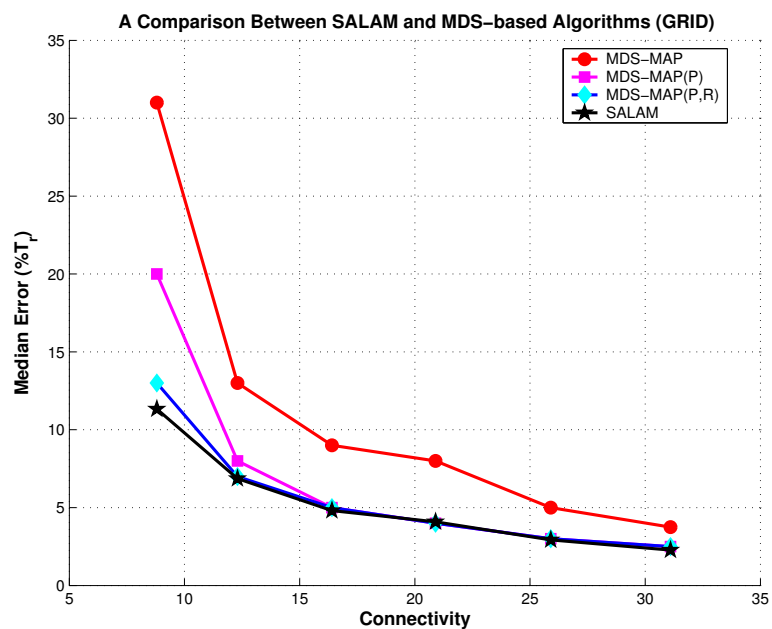


Figure 6.11: A comparison between SALAM and MDS-based algorithms using GRID topology

We also compared the accuracy achieved by SALAM against the accuracy of centralized range-free algorithms. SALAM is much better than the convex optimization approach [37] when the number of anchor nodes is low. For example, with 4 to 10 anchors in a 200-node random network, the convex optimization approach has an average estimation error of more than twice the transmission range, when the transmission range is 12.5 and above (i.e. more than 25 units of distance).

SALAM is also better than Hop-TERRAIN [74], especially when the number of anchors is small. For example, with 3 anchors (2%) and a average node degree 12, SALAM has an average error of about $10\%T_r$, whereas Hop-TERRAIN has an average error of about $90\%T_r$.

Chapter 7

Conclusions and Research Directions

Sensor networks classify as ad-hoc networks with large number of nodes and limited power and computational capacities. With ad-hoc deployment one cannot accurately predict or plan a-priori the location of each sensor node. Moreover, using GPS is not always a suitable solution. These unique features have raised some interesting challenges that must be considered when designing a localization algorithm for sensor networks. The protocol must be scalable, power efficient, GPS-free, and still achieves acceptable accuracy.

In this dissertation, we have presented SALAM, a scalable GPS-free range-based localization algorithm for wireless sensor networks. SALAM assumes that each node has the capability to estimate *ranges* (distances) to its corresponding neighbors, that are within its transmission range, with some error. We laid out a taxonomy of the current research in the area of ad-hoc location determination systems and showed where SALAM belongs in this taxonomy.

Scalability is achieved through grouping sensors into overlapping multi-hop clusters. Clustering facilitates the distribution of control over the network and, hence, enables locality of communication. Clustering nodes into groups saves energy and reduces net-

work contention because nodes communicate their data over shorter distances to their respective cluster heads instead of network-wide flooding.

Each cluster head is responsible for building a local relative map corresponding to its cluster using intra-cluster node's range measurements. To obtain the global relative topology of the network, the cluster head nodes collaboratively combine their local maps using simple matrix transformations.

In order for two cluster heads to perform these matrix transformations, the two clusters must be overlapping with degree at least 3. We formulated the overlapping multi-hop clustering problem as an extension to the k -dominating set (KDS) problem. Since the problem is NP-Hard, we introduced the *OK* randomized multi-hop heuristic algorithm for solving it. *OK* is scalable in terms of communication overhead and terminates in a constant number of iterations independent of the network size.

We studied the characteristics of *OK* through analytical analysis and simulation. *OK* parameters, such as cluster radius, average node degree, and cluster head probability can be easily tuned to achieve the application design goals with high probability. The results showed that with high probability *OK* provides high network coverage and connectivity. Moreover, by selecting the parameter values we can achieve a certain average overlapping degree and control the cluster size. Although *OK* generates overlapping clusters, the simulation results show that the clusters are approximately equal in size. This is desirable to achieve load balancing between different clusters. We have developed a detailed analytical model and have shown that it is valid by comparison with the simulation results.

A major problem with intra-cluster (local) location discovery is the error accumu-

lated in the node position as it becomes multi-hop away from the cluster head node. We analyzed different sources of error and developed techniques to avoid those errors. We designed and implemented the Multi-hop Relative Location Estimation (MRLE) algorithm that uses these heuristics to estimate relative node's positions with low error margins. For higher accuracy, we use an optional refinement step, where we iteratively improve the initial position estimate by formulating a least-squares metric and solving it using non-linear optimization techniques. By using the optional refinement phase, we give the sensor network engineer a tool to trade between computational power and accuracy.

We showed how the local coordinate system (LCS) affects the accuracy of the estimated position dramatically and we proposed different heuristics to select the LCS and compare between these heuristics in terms of accuracy and time complexity. The results show that the minimum initial error (MIE) heuristic can estimate an initial node position that is very close to the optimal position. However, the MIE heuristic require more computational overhead compared with other heuristics; hence, allowing the application to trade computational power for accuracy. We have shown that we can avoid the computationally expensive optimization problem by spending some time in selecting the coordinate system.

We analyzed the accuracy of the intra-cluster location discovery via simulation. We captured the impact of the different parameters, such as cluster radius and connectivity on the accuracy of the estimated position. We introduced a new metric, the CLIQUE factor. Our experiments have concluded that the CLIQUE factor has a very dominant effect on the estimation accuracy regardless of the cluster size. We showed that we can trade trade the accuracy of the estimated position against node transmission range; hence,

the application layer can choose from a whole range of different options, to estimate the sensor nodes' positions with different accuracy while conserving battery power.

We also analyzed the accuracy of the inter-cluster (global) location discovery. We introduced a new problem, the *best order of transformations* between clusters. We formulated the problem as finding a spanning tree for the *overlapping graph*. We proposed different heuristics to assign weights to the edges of the overlapping graph in order to minimize the inter-cluster error and minimize the communication overhead per node. We also proposed two approaches to construct the spanning tree of the overlapping graph: minimum spanning tree and minimum height spanning tree.

Simulation results show that the spanning tree of the overlapping graph highly affects both accuracy and communication overhead of the system. The minimum height spanning tree always leads not only to better communication overhead, but also better accuracy since the number of transformations is reduced. We also captured the impact of the overlapping degree between clusters on the accuracy of the estimated node's positions.

We compared the performance of SALAM to the performance of MDS-based techniques. We showed that SALAM is more accurate than both the MDS-MAP system and the enhanced MDS-MAP system by more than $30\%T_r$, $15\%T_r$ respectively, at low node connectivity. We also compared SALAM against centralized range-free algorithms [37]. With 4-10 anchor nodes, the convex optimization approach has an average estimation error of more than 25 units of distance, however, SALAM has an average median error of less than 1 unit distance when the ranges have error with standard deviation 4. Finally, we compared the accuracy of SALAM to the accuracy of range-based anchor-based algorithms (HOP-TERRAIN), we showed that with 3 anchors, SALAM has an average error

of about $10\%T_r$, whereas Hop-TERRAIN has an average error of about $90\%T_r$.

To conclude, we showed in this research work that locally centralized algorithms scale well with increased network size; yet, they can achieve acceptable accuracy compared to a centralized approach. We showed that a locally centralized algorithm is indeed a good compromise between accuracy, communication overhead. Although we analyzed the performance of SALAM in the context of wireless sensor networks, SALAM is applicable for general ad-hoc networks.

7.1 Research Directions

1. **Finding absolute position.** In order to find the absolute position of the nodes, some sensor nodes must be GPS-enabled. SALAM currently assumes only three anchor nodes in order to find the absolute nodes' positions. In the future research, we plan to study the effect of increasing the number of GPS-enabled nodes on accuracy, and where to place them.
2. **Mobility.** In the current implementation of SALAM, the sensor nodes are assumed to be stationary, which is a valid assumption, for sensor networks. The mobility of the some nodes can be desirable in numerous applications. For example, an emergency vehicle equipped with computing and communication devices in the context of a disaster management application and a walking soldier with a laptop computer in his backpack in a battle environment. However, as the nodes moves around, the relative node positions need to be recomputed. As a future research direction, we plan to study the power consumption due to node mobility.

3. **Edge-based Clustering.** Almost all clustering algorithms divide the network such that all clusters have the same number of nodes. However, in SALAM, we have shown how the CLIQUE factor affects accuracy. In order to achieve a certain accuracy, the number of edges per cluster should be above a certain threshold. This indicate that the clustering should be edge-based not node-based. As a future work, we plan to design a clustering algorithm that divides the network into clusters such that the CLIQUE factor per cluster is greater than a certain threshold.
4. **Cluster Maintenance.** In the current implementation of SALAM, we assume static nodes. Hence, SALAM runs once after the network bootstrapping to estimate nodes' positions and terminates. Therefore, we do not analyze the performance in case of node failure (cluster head or non-cluster head nodes). However, if mobility is to be considered, we need to investigate the behavior of the proposed algorithm in the event of sensor failures.
5. **Load Balancing.** In general, cluster head nodes spend relatively more energy than other sensors because they have to receive information from all the sensors within their cluster. Hence, they may run out of their energy faster than other sensors. In the current implementation, the clustering algorithm runs once during network bootstrapping. Hence, there is no need to share the cluster head role among all nodes. However, if mobility is to be considered, it is necessary to switch the cluster head role between nodes in order to maximize the network life time. One possible solution is to run the clustering algorithm periodically for load balancing. Another possibility is that cluster heads trigger the clustering algorithm when their energy

levels fall below a certain threshold. We leave this as a future extension to the OK protocol.

6. **More Accurate Analytical Model.** In the analytical model, we assume circle representation for the cluster. As a future extension, we may want to look into different improvements to derive a tighter bound by studying the actual cluster shape using more complex stochastic geometry techniques.

Appendix A

Area of Intersection Between Two Identical Circles

Assume that we have two identical circles A and B that intersect in some area I_{AB} . Let r be the radius length and w be the distance between the two centers A and B as shown in

Fig. A.1. then the intersection (I_{AB}) can be calculated as follows:

$$I_{AB} = 2 (\text{area of sector } CBD - \text{area of triangle } CBD)$$

$$\text{Area of sector } CBD = \frac{1}{2} \cdot 2\theta \cdot R^2 = \theta \cdot R^2$$

$$\therefore I_{AB} = 2(\theta R^2 - \frac{1}{2} \cdot R^2 \sin 2\theta) = (2\theta - \sin 2\theta)R^2$$

where $w = 2R \cos \theta$ (using cosine rule)

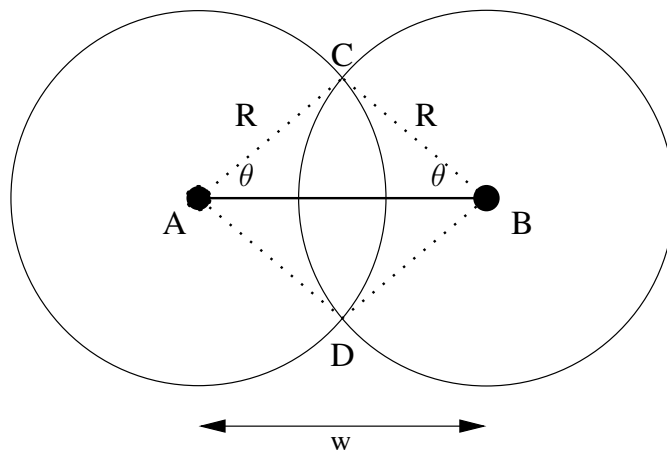


Figure A.1: Area of intersection of two circles

BIBLIOGRAPHY

- [1]
- [2] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E.Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.
- [3] K.M. Alzoubi, P.-J. Wan, and O. Frieder. Distributed Heuristics for Connected Dominating Sets in Wireless Ad Hoc Networks. *Journal of Communications and Networks*, 4(1), March 2002.
- [4] K.M. Alzoubi, P.-J. Wan, and O. Frieder. Message-Optimal Connected Dominating Sets in Mobile Ad Hoc Networks. In *MOBIHOC*, EPFL Lausanne, Switzerland, June 2002.
- [5] K.M. Alzoubi, P.-J. Wan, and O. Frieder. New Distributed Algorithm for Connected Dominating Set in Wireless Ad Hoc Networks. In *Proceedings of the 35th Hawaii International Conference on System Sciences*, Big Island, Hawaii, 2002.
- [6] A.D. Amis and R. Prakash. Load-Balancing Clusters in Wireless Ad Hoc Networks. In *Proceedings of ASSET*, Richardson, Texas, March 2000.

- [7] Alan D. Amis, Ravi Prakash, Thai H. P. Vuong, and Dung T. Huynh. Max-Min D-Cluster Formation in Wireless Ad Hoc Networks. In *IEEE INFOCOM*, March 2000.
- [8] D. J. Baker and A. Ephremides. The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm. *IEEE Transactions on Communications*, 29(11):1694–1701, November 1981.
- [9] Seema Bandyopadhyay and Edward Coyle. An Energy-Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks. In *IEEE INFOCOM*, San Francisco, CA, March 2003.
- [10] S. Banerjee and S. Khuller. A clustering scheme for hierarchical control in multi-hop wireless networks. In *IEEE INFOCOM*, 2001.
- [11] S. Basagni. Distributed and Mobility-Adaptive Clustering for Multimedia Support in Multi-Hop Wireless Networks. In *Proceedings of Vehicular Technology Conference*, volume 2, pages 889–893, 1999.
- [12] S. Basagni. Distributed Clustering for Ad Hoc Networks. In *Proceedings of International Symposium on Parallel Architectures, Algorithms and Networks*, pages 310–315, June 1999.
- [13] J. Beal. A robust amorphous hierarchy from persistent nodes. *AI Memo*, (11), 2003.
- [14] Jeremy Blum, Min Ding, Andrew Thaeler, and Xiuzhen Cheng. *Handbook of Combinatorial Optimization*. Kluwer Academic Publishers, 2004.

- [15] K. S. Booth and J. H. Johnson. Dominating sets in chordal graphs. *SIAM J. Comput.*, 11:191–199, 1982.
- [16] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less Low-cost Outdoor Localization for Very Small Devices. *IEEE Personal Communications*, 7(5):28–34, October 2000.
- [17] Nirupama Bulusu, J. Heidemann, V. Bychkovskiy, and D. Estrin. Density-adaptive beacon placement algorithms for localization in ad hoc wireless networks. In *IEEE Infocom 2002*, June 2002.
- [18] S. Butenko, X. Cheng, C. Oliveira, and P.M. Pardalos. *Recent Developments in Cooperative Control and Optimization*. Kluwer Academic Publishers, 2004.
- [19] S. Butenko, C. Oliveira, and P.M. Pardalos. A new algorithm for the minimum connected dominating set problem on ad hoc wireless networks. In *Proceedings of CCCT'03*, pages 39–44, 2003.
- [20] M. Cadei, X. Cheng, and D.-Z. Du. Connected Domination in Ad Hoc Wireless Networks. In *Proc. 6th International Conference on Computer Science and Informatics*, 2002.
- [21] S. Capkun, M. Hamdi, and J.-P. Hubaux. Gps-free positioning in mobile adhoc networks. In *in Hawaii International Conference on System Sciences (HICSS-34)*, pages 3481–3490, January 2001.
- [22] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. In *In ACM SIG-*

COMM Workshop on Data Communications in Latin America and the Caribbean,
April 2001.

- [23] A. Cerpa and D. Estrin. ASCENT: Adaptive Self-Configuring Sensor Networks Topologies. In *Proceedings of IEEE INFOCOM*, New York, NY, June 2002.
- [24] G. J. Chang and G. L. Nemhauser. The k-domination and k-stability problem on graphs. Technical Report 540, School of Operations Research and Industrial Engineering, Cornell University, 1982.
- [25] M. Chatterjee, S. K. Das, and D. Turgut. WCA: A Weighted Clustering Algorithm for Mobile Ad hoc Networks. *Journal of Cluster Computing, Special issue on Mobile Ad hoc Networking*, (5):193–204, 2002.
- [26] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: an Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. *ACM Wireless Networks*, 8(5), September 2002.
- [27] Y. Chen and A. Liestman. Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks. In *MOBIHOC*, EPFL Lausanne, Switzerland, June 2002.
- [28] X. Cheng, M. Ding, and D. Chen. An approximation algorithm for connected dominating set in ad hoc networks. In *Proc. of International Workshop on Theoretical Aspects of Wireless Ad Hoc, Sensor, and Peer-to-Peer Networks (TAWN)*, 2004.

- [29] X. Cheng, M. Ding, D.H. Du, and X. Jia. On The Construction of Connected Dominating Set in Ad Hoc Wireless Networks. *Special Issue on Ad Hoc Networks of Wireless Communications and Mobile Computing*, 2004.
- [30] X. Cheng, A. Thaeler, G. Xue, and D. Chen. Tps: A time-based positioning scheme for outdoor sensor networks. In *in the Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, March 2004.
- [31] C.F. Chiasserini, I. Chlamtac, P. Monti, and A. Nucci. Energy Efficient design of Wireless Ad Hoc Networks. In *Proceedings of European Wireless*, February 2002.
- [32] L. Clare, G. Pottie, and J. Agre. Self-organizing distributed sensor networks. In *SPIE Conf. Unattended Ground Sensor Technologies and Applications*, pages 229–237, Orlando, FL, April 1999.
- [33] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86:165–177, 1990.
- [34] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. MIT Press and McGraw-Hill, 2001.
- [35] B. Das and V. Bharghavan. Routing in Ad-Hoc Networks Using Minimum Connected Dominating Sets. In *ICC*, 1997.
- [36] Murat Demirbas, Anish Arora, and Vineet Mittal. FLOC: A Fast Local Clustering Service for Wireless Sensor Networks. In *1st Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks*, Florence, Italy, June 2004.

- [37] L. Doherty, K. Pister, and L. El Ghaoui. Convex position estimation in wireless sensor networks. In *in the Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, April 2001.
- [38] A. Ephremides, J.E. Wieselthier, and D. J. Baker. A Design concept for Reliable Mobile Radio Networks with Frequency Hopping Signaling. *Proceeding of IEEE*, 75(1):56–73, 1987.
- [39] A. Mainwaring et al. Wireless sensor networks for habitat monitoring. In *In ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, September 2002.
- [40] H. Wang et al. Target classification and localization in habitat monitoring. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2003)*, April 2003.
- [41] S. Meguerdichian et al. Coverage problems in wireless ad-hoc sensor networks. In *in the Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, April 2001.
- [42] T. He et al. Range-free localization schemes for large scale sensor networks. In *in the Proceedings of the ACM Conference on Mobile Computing and Networks (MOBICOM'03)*, September 2003.
- [43] M. Faloutsos and M. Molle. Optimal distributed algorithm for minimum spanning trees revisited. In *Proceedings of the Fourteenth Annual ACM Symposium on Principles of Distributed Computing*, 1995.

- [44] M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness*. Freeman, San Francisco, 1978.
- [45] S. Ghiasi, A. Srivastava, X. Yang, and M. Sarrafzadeh. Optimal Energy Aware Clustering in Sensor Networks. *Sensors Magazine*, (1):258–269, January 2002.
- [46] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, April 1998.
- [47] G. Gupta and M. Younis. Load-Balanced Clustering in Wireless Sensor Networks. In *the International Conference on Communication (ICC 2003)*, Anchorage, Alaska, May 2003.
- [48] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater. *Domination in Graphs: Advanced Topics*. Marcel Dekker, Inc. New York, 1998.
- [49] W. B. Heinzelman, A. P. Chandrakasan, , and H. Balakrishnan. An Application Specific Protocol Architecture for Wireless Microsensor Networks. *IEEE Transactions on Wireless Networking*, 1(4), October 2002.
- [50] M. A. Henning, O. R. Oellermann, and H. C. Swart. The diversity of domination. *Discrete Mathematics*, 161(3):161–173, December 1996.
- [51] M. A. Henning, O.R. Oellermann, and H.C. Swart. Bounds on distance domination parameters. *J. Combin. Inform. System Sci.*, 16:11–18, 1991.
- [52] Jeffrey Hightower and Gaetano Borriella. Location Systems for Ubiquitous Computing. *IEEE Computer*, 34(8):57–66, August 2001.

- [53] B. Horn, H.M. Hilden, and S. Negahdaripour. Closed form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of America*, 5:1127–1135, 1998.
- [54] A. Howard, M.J. Mataric, and G.S. Sukhatme. Relaxation on a mesh: a formalism for generalized localization. In *in Proc. of the IEEE International Conference on Intelligent Robots and Systems (IROS01)*, pages 1055–1060, 2001.
- [55] Steven Huss-Lederman, Elaine M. Jacobson, Anna Tsao, Thomas Turnbull, and Jeremy R. Johnson. Implementation of strassen’s algorithm for matrix multiplication. In *Supercomputing '96: Proceedings of the 1996 ACM/IEEE conference on Supercomputing (CDROM)*, Washington, DC, USA, 1996.
- [56] C. Intanagonwivat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *in the Proceedings of the ACM Conference on Mobile Computing and Networks (MOBICOM)*, pages 56–67, September 2000.
- [57] X. Ji. Sensor positioning in wireless ad-hoc sensor networks with multidimensional scaling. In *in the Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, March 2004.
- [58] L.R. Ford Jr. and D.R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.
- [59] V. Kawadia and P. R. Kumar. Power Control and Clustering in Ad Hoc Networks. In *IEEE INFOCOM*, San Francisco, CA, March 2003.

- [60] T. Kim and V. Bharghavan. Multicast routing in heterogeneous wireline/wireless environments. In *IEEE Wireless Communications and Networking Conference*, September 1999.
- [61] B. Krishnamachari, S. Wicker, and R. Bejar. Phase Transition Phenomena in Wireless Ad-hoc Networks. In *in GLOBECOM*, San Antonio, TX, 2001.
- [62] T. J. Kwon and M. Gerla. Clustering with Power Control. In *Proceeding of MILCOM99*, 1999.
- [63] K. Langendoen and N. Reijers. Distributed Localization in Wireless Sensor Networks: A Quantitative Comparison. *Computer Networks (Elsevier)*, special issue on *Wireless Sensor Networks*, pages 374–387, August 2003.
- [64] L. Li and J.Y. Halpern. Minimum-energy mobile wireless networks revisited. In *IEEE International Conference on Communications*, June 2001.
- [65] C. R. Lin and M. Gerla. Adaptive Clustering for Mobile Wireless Networks. *Journal on Selected Areas in Communication*, 15:1265–1275, September 1997.
- [66] A. B. McDonald and T. Znati. A Mobility Based Framework for Adaptive Clustering in Wireless Ad-Hoc Networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1466–1487, August 1999.
- [67] J. C. Navas and T. Imielinsk. Geographic addressing and routing. In *in the Proceedings of the ACM Conference on Mobile Computing and Networks (MobiCom)*, September 1997.

- [68] D. Niculescu and B. Nath. Ad-hoc positioning system. In *in the Proceedings of IEEE Global Communication Conference (Globcom'01)*, November 2001.
- [69] D. Niculescu and B. Nath. Ad hoc positioning system (aps) using aoa. In *in the Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, March 2003.
- [70] A. K. Parekh. Selecting Routers in Ad-Hoc Wireless Networks. In *Proceedings of ITS*, 1994.
- [71] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, April 2000.
- [72] T. Rappaport. *Wireless Communications: Principles & Practice*. Englewood Cliffs, NJ: Prentice-Hall, 1996.
- [73] S.I. Roumeliotis and G.A. Bekey. Synergetic localization for groups of mobile robots. In *in Proc. of the 39th IEEE Conference on Decision and Control*, page 34773482, December 2000.
- [74] C. Savarese, J. Rabaey, and K. Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In *USENIX Technical Annual Conference*, June 2002.
- [75] A. Savvides, C. C. Han, and M. Srivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *in the Proceedings of the ACM Conference on Mobile Computing and Networks (MOBICOM'01)*, July 2001.

- [76] A. Savvides, H. Park, and M. B. Srivastava. The bits and flops of the n-hop multilateration primitive for node localization problems. In *in the Proceedings of the first ACM international workshop on Wireless Sensor Networks and Applications*, September 2002.
- [77] Andreas Savvides and M. B. Srivastava. The N-Hop Multilateration Primitive for Node Localization Problems. *ACM MONET special issue on Wireless Sensor Networks and Applications*, 2003.
- [78] Loren Schwiebert, Sandeep K. S. Gupta, and Jennifer Weinmann. Research challenges in wireless networks of biomedical sensors. In *in the Proceedings of the ACM Conference on Mobile Computing and Networks (MOBICOM'01)*, pages 151–165, July 2001.
- [79] Y. Shang and W. Ruml. Improved mds-based localization. In *in the Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, March 2004.
- [80] Yi Shang, Wheeler Ruml, Ying Zhang, and Markus P. J. Fromherz. Localization From Mere Connectivity. In *In Proc. of ACM MOBIHOC 2003*, pages 201–212, Annapolis, MD, June 2003.
- [81] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In *in the Proceedings of the ACM Conference on Mobile Computing and Networks (MOBICOM'01)*, pages 272–286, July 2001.

- [82] S. Skiena. *Handbook of Combinatorial Optimization*. Reading, MA: Addison-Wesley, 1990.
- [83] Mani Srivastava, Richard Muntz, and Miodrag Potkonjak. Smart kindergarten: Sensor-based wireless networks for smart developmental problem-solving environments. In *in the Proceedings of the ACM Conference on Mobile Computing and Networks (MOBICOM'01)*, pages 132–138, July 2001.
- [84] I. Stojmenovic and X. Lin. Loop-free Hybrid Single-path Flooding Routing Algorithms with Guaranteed Delivery for Wireless Networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1023–1032, October 2001.
- [85] P.-J. Wan, K.M. Alzoubi, and O. Frieder. Distributed Construction of Connected Dominating Sets in Wireless Ad Hoc Networks. In *IEEE INFOCOM*, 2002.
- [86] Y. Xu, J. Heidemann, and D. Estrin. Geography-Informed Energy Conservation for Ad Hoc Routing. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, pages 70–84, Rome, Italy, July 2001.
- [87] F. Ye, H. Luo, J. Chung, S. Lu, and L. Zhang. A Two-Tier Data Dissemination Protocol for Large-Scale Wireless Sensor Networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, Atlanta, Georgia, September 2002.

- [88] Ossama Younis and Sonia Fahmy. Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach. In *IEEE INFOCOM*, Hong Kong, March 2004.
- [89] A. Youssef, A. Agrawala, and M. Younis. Accurate Anchor-Free Localization in Wireless Sensor Networks. In *in the Proceedings of the 1st IEEE Workshop on Information Assurance in Wireless Sensor Networks (WSNIA 2005)*, Phoenix, Arizona, April 2005.