Robust Networking in Dense Wireless Networks

By

Theodore Lee Ward, MS/BS

Bachelor of Science in Computer Science
Oklahoma State University
Stillwater, Oklahoma
1992

Master of Science in Computer Science
University of Tulsa
Tulsa, Oklahoma
2006

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment
of the requirements for
the degree of
DOCTOR OF PHILOSOPHY
July 2015

Name: THEODORE WARD

Date of Degree: JULY, 2015

Title of Study: ROBUST NETWORKING IN DENSE WIRELESS NETWORKS

Major Field: COMPUTER SCIENCE

Abstract: Cyber Physical Systems (CPS) and Nanotechnology are emerging fields that promise to unite the physical and computing realms through the use of impossibly tiny sensor based devices called *motes*. The revolutionary nature of these systems stems from the unique properties of these mote devices such as their use of sensors, small size and low cost per unit.

These CPS networks are expected to control real time critical applications requiring deployment in remote and hostile environments in which far reaching communication or constant human oversight can not be assumed. For this reason some of the defining characteristics of CPS are the ability to be self-configuring, self-policing and fault tolerant. One area of concern is wireless communication technology, existing methods have been deemed inadequate as a foundation for cyber physical systems.

We present a distributed wireless communication strategy that can achieve a probabilistic degree of *logical redundancy* of communication between the motes that sense information and the data sinks that consume the information. We can achieve this by taking advantage of the expected low cost and small size of motes in order to assume the inevitability of, or ability to deploy, very dense mote populations. Our strategy then is to partition the motes in a densely populated CPS network into multiple overlapping logical networks (LNETs). Each LNET operates as an independent CPS and thus if some event is sensed by motes in n separate LNETs, then accounts of the event will be reported with redundancy equal to $n - 1$.

Our simulations show that as population density fluctuates, partitioning in this manner provides **capacity scaling** by expanding bandwidth and decreasing resource contention, and **increased reliability** of communication due to logical redundancy.

Contents

Abstract iii

List of Figures x

1 Introduction **1**

1.1 Cyber Physical System 1

1.2 Problem Statement 2

1.3 Definitions 4

1.3.1 List of symbols 5

1.4 Research Goals and Motivation 7

1.5 Issues identified and how they are addressed 8

1.5.1 Applicability to cyber physical systems 10

1.5.2 Scalability 10

1.5.3 Cyber security 10

1.6 Contribution 12

1.7 Organization 12

2 Literature Review **14**

2.1 Wireless communication 14

2.2 Density 15

2.2.1 Highly dense wireless networks 17

2.2.2 Probabilistic connectivity 18

2.2.3 Edge effect 18

2.3 Coverage and Partitioning 20
2.3.1 Partitioning methods 21
2.4 Deployment 22
2.5 Multi-channel 23
2.6 Simulation models 24

3 Density **26**

3.1 Introduction 26
3.1.1 Motivation 27
3.1.2 Definitions 27
3.1.3 Metrics 30
3.1.4 Contribution 31
3.2 Measuring density and redundancy 31
3.2.1 Distributed density measurement 32
3.3 Base density and density factors 34
3.3.1 Redundancy factor 34
3.4 The edge effect 36
3.4.1 Countering edge effect 39
3.4.2 Effective density 45
3.5 Proactive density manipulation 47
3.6 Conclusion 49

4 Partitioning **50**

4.1 Introduction 50
4.1.1 Motivation 51
4.1.2 Definitions 51
4.1.3 Metrics 52
4.1.4 Contribution 52

4.2 Partitioning approaches 53
4.2.1 Metric for ranking neighbors 53
4.2.2 Random 53
4.2.3 Offset grid 54
4.2.4 Mutually exclusive neighbors 55
4.2.5 Turn taker partitioning algorithm 56
4.3 Proactive redundancy factor manipulation 59
4.4 Conclusion 60

5 Protocol **61**
5.1 Overview 61
5.1.1 Motivation 61
5.1.2 Definitions 62
5.1.3 Contribution 64
5.2 The LNET protocol 64
5.2.1 Micro state and macro state 65
5.2.2 Coordination of motes 66
5.2.3 Triggering the LNET phase 66
5.2.4 The LNET initiation process 68
5.3 States 70
5.3.1 Macro assessment 70
5.3.2 Channel survey 72
5.3.3 Neighbor discovery 73
5.3.4 Density calculation 75
5.3.5 Partitioning 75
5.4 Conclusion 77

6 Simulation **78**

6.1 Simulation . . . 78
6.1.1 Purpose . . . 79
6.1.2 Existing simulation packages . . . 79
6.2 Simulator models . . . 81
6.2.1 Code structure . . . 81
6.2.2 Event model . . . 82
6.2.3 Distribution model . . . 83
6.2.4 Failure model . . . 83
6.2.5 Contention model . . . 84
6.2.6 Radio configuration and modeling . . . 84
6.2.7 Routing model . . . 85
6.3 Simulation control model . . . 85
6.3.1 Ideal density . . . 86
6.4 Simulation of redundancy . . . 87
6.4.1 Single channel . . . 87
6.4.2 Multiple channels, independent partitions . . . 88
6.4.3 Multiple channels, redundant partitions . . . 90
6.5 Full protocol simulation . . . 91
6.6 Conclusion . . . 91

7 Conclusion **94**
7.1 Further research . . . 94
7.1.1 Optimizations . . . 95
7.1.2 Extension . . . 95

Appendices **97**

A Definitions **98**
A.1 Terms . . . 98

A.2 Symbols 99
A.3 Time periods 100

Bibliography **101**

List of Figures

1.1 Identified needs and requirements 3

1.2 Addressed needs and requirements 4

1.3 Two networks communicating an event redundantly to a sink. 8

1.4 How needs are addressed 9

1.5 Use of redundancy to detect malicious data at sink 11

1.6 Chapter descriptions 13

3.1 Motescapes with varying densities. 27

3.2 Redundant coverage 35

3.3 The edge effect 37

3.4 Mutually exclusive neighbors of center, edge and corner motes. 38

3.5 Motes excluded from density calculation by statistical estimation. . . 41

3.6 Odd shaped network 42

3.7 Deviation from $E(\delta)$ when varying field size. 44

3.8 Impact of field size on edge 48

4.1 Definition of partition 51

4.2 Desirable attributes of a partition 52

4.4 Perfect distribution of neighbors. 55

4.5 Mote 1 and mutually exclusive neighbors 2,3,4,5. 56

4.6 Mutually exclusive neighbors chosen by center node. 57

4.7 Turn-taking partitioning algorithm 57

4.8 Density measurements when creating three partitions. 58

4.9 Density measured in partitions created using turn taker and Random partitioning methods. 59

5.1 States and transitions. 63

5.2 Turn-taking distributed implementation 76

6.1 Structure of simulator code 82

6.2 Radio configuration. 85

6.3 Varying density to determine an ideal. 87

6.4 Single channel as density rises. 88

6.5 Multiple channels as density rises 89

6.6 Redundant networks over multiple channels 90

6.7 LNET scaling in response to population increase 92

6.8 Breakout of statistics from full simulation. 92

Chapter 1

Introduction

1.1 Cyber Physical System

Many fields within computer science incorporate the idea of networks of tiny computing devices. Some of the terms coined in this regard are smart dust, cyber physical systems, programmable matter, ubiquitous computing, the Internet of things, utility fog and claytronics[1, 2, 3]. These terms refer to similar concepts albeit with different applications or hailing from different scientific domains. In this text we adhere to the notion and nomenclature of Cyber Physical Systems (CPS), a burgeoning field of research in computer science identified by the US National Science Foundation as a key research priority in 2006[4].

The emerging field of cyber physical systems is jointly physical and computational[5]. a Cyber Physical System (CPS) is a system incorporating numerous tiny wirelessly connected sensing devices called *motes* and devices that can manipulate physical environment called *actuators*. A CPS will utilize its motes and actuators to learn from and and manipulate its surroundings, bridging the gap between computing and physical realms. The feasibility of tiny wireless motes was demonstrated in 1998

when a group of researchers from the University of California, Berkeley demonstrated a working mote smaller than a grain of rice[6].

Since CPS is an emerging field its definition, scope and requirements are somewhat fluid. Key attributes necessary for full scale CPS adoption such as architecture, price point, size, and adequate power sources remain inadequate, technologically infeasible or commercially non-viable[5, 7]. In an examination of CPS foundations Lee claims that until it is feasible to make wireless links predictable and reliable we must compensate with "robust coding schemes and adaptive protocols"[3, 8].

The individual motes in a CPS will have limited capability by design but due to low cost, large populations and small size these systems could "dwarf the 20th century IT revolution" [5]. One reason that cyber physical systems have such great expectations is due to their intended application. Groundbreaking uses for a CPS will be critical applications necessary to keep us safe and sustain or improve our quality of life [9]. These applications include management of SCADA systems such as power and water, coordination of air traffic and navigation of automobiles. A single collection of motes may be composed of thousands, millions, possibly even trillions of motes[10, 11].

1.2 Problem Statement

Cyber physical systems are becoming integral to applications within vital sectors such as military, infrastructure and transportation[12, 13]. Before these systems can be fully adopted a number of foundational issues need to be addressed, these were enumerated in 2012 by Edward A. Lee of UC Berkeley with input from other prominent figures in the CPS community. Beginning with a taxonomy given at a National Institute of Standards (NIST) workshop on CPS, Lee created the concept map in figure 1.1. This concept map provides defining characteristics of a CPS, describes some

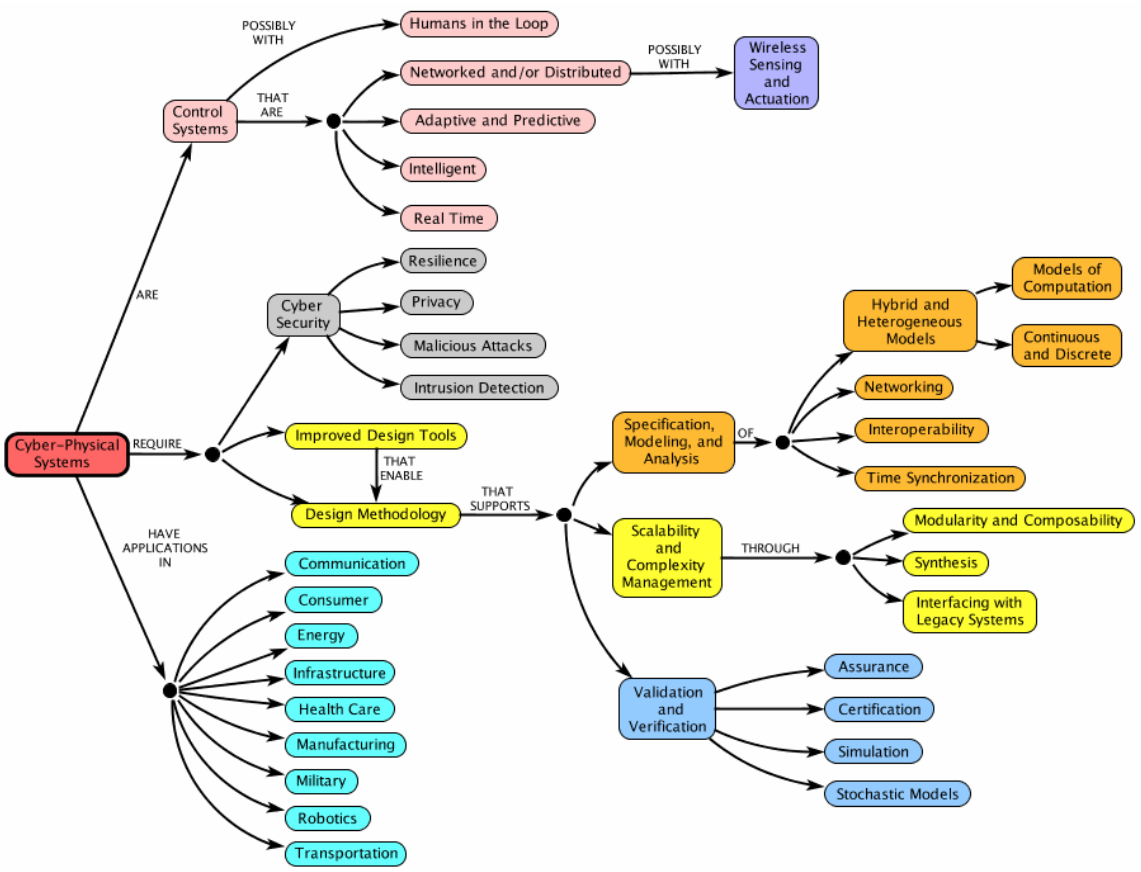


Figure 1.1: Concept map from http://cyberphysicalsystems.org

important CPS applications, it also identifies needs from foundational papers such as Lee's own "Cyber Physical Systems: Are computing foundations adequate?"[5] and Derler's "Modeling Cyber Physical Systems"[14] that are necessary for solid foundations on which to build next generation Cyber Physical Systems.

Cyber security problems involve the ability to be resilient in the face of malicious attacks, an especially worrisome issue in CPSs that are remotely distributed and not monitored by humans. An important part of cyber security is the ability to detect attacks in order to prevent them from succeeding, or to mitigate the damage caused.

Scalability of network communication is particularly important because a CPS may incorporate dynamic population changes due to the addition of motes, or because motes exhaust their power sources.

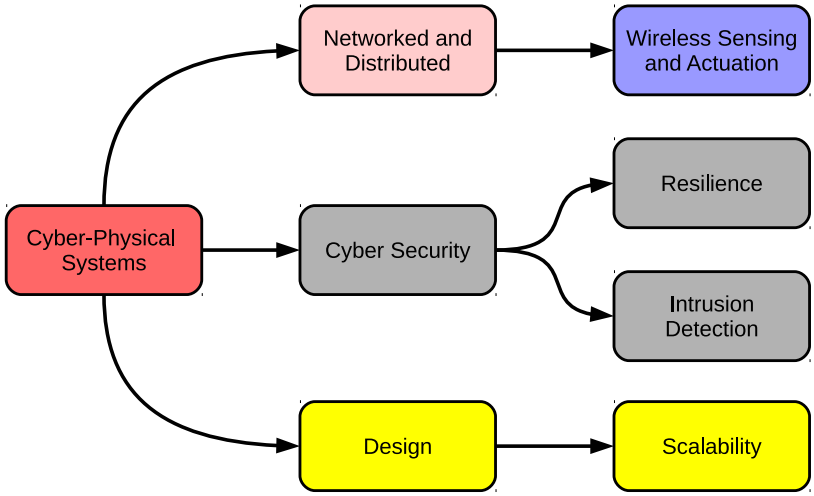


Figure 1.2: Concepts addressed within this dissertation

The research in this dissertation is aimed at improving cyber-security resilience and providing scalability in a CPS considering a distributed environment comprised of a large population of low power devices. Our work may also be useful in facilitating a means of intrusion detection.

1.3 Definitions

In this section we give a brief overview of wireless communication and define many of the terms used in this dissertation.

Radio communication Motes communicate using low power radio waves that are modulated to represent binary data. A mote that wishes to send a message will act as a radio transmitter and broadcast its modulated signal within a given *frequency range*. Motes that are near enough to receive this broadcast and whose radios are tuned to the correct frequency will receive the message and demodulate the signal to retrieve the data. If two motes are within range of one another and tuned to the

same radio frequency the are called *neighbors* and the *degree* of a mote refers to the number of neighbors a mote has.

Channels Invariably there exists a set of radio frequency ranges that can be used for transmitting data, each range is referred to as a *channel*. The channels available for use are defined by the communication protocol and often varies by country or region. The channel used by motes for communication depends on various factors but often comes down to avoiding congestion or interference. *Congestion* occurs whenever too many neighboring motes attempt to transmit data simultaneously causing some motes to postpone transmission, extreme congestion may result in failure to transmit. A signal is counted as *interference* if the mote receiving it is unable to decode it to produce a properly formatted packet.

Transmission range A radio broadcast is omni-directional, so when a mote broadcasts a message it is at the center of a virtual circle or sphere defined by the transmission power of the radio. The *radius* r_0 is the distance from the transmitter at which the broadcast can be received. The area around a mote within which a transmission can be received is referred to as the *unit area* and for two dimensions is calculated as $U_r = \pi r_0^2$

Density The density of a motescape is the average number of motes per unit of area within its borders and is defined in equation 3.1 of chapter 3.

1.3.1 List of symbols

The symbols used throughout this text presented with a brief definition.

Symbols

- F A field of motes or *motescape*.
- A The area of F.
- M The set of all motes that populate F.
  - m Some mote $m \in M$.
  - $|M|$ The size of set M, or the number of motes M contains.
- s_0 The radius of a mote's sensor range.
- r_0 The radius of a mote's radio communication range.
- U The unit of area against which density values are calculated.
  - U_1 Base unit of area, $U = 1$.
  - U_s Unit of area equal to sensor coverage area $U = \pi s_0^2$.
  - U_r Unit of area equal to radio coverage area $U = \pi r_0^2$.
- $\deg(m)$ The degree of a mote m or the number of one hop neighbors m has.
- δ Density, the calculated population size per unit of area.
  - $\delta(U)$ Function to calculate density with respect to unit of area U.
  - $\delta_1, \delta_r, \delta_s$ Shorthand for $\delta(U_1)$, $\delta(U_r)$, and $\delta(U_s)$ respectively.
  - $\hat{\delta}$ An approximation of δ.
- μ The mean degree observed in a set of motes.
  - $\mu(S)$ Algorithm for computing the mean degree of motes in set S.

  - μ_r, μ_s Mean degree of devices that have a unit of area equal to U_r, and U_s respectively.

- ℓ Ideal degree, the mote degree that we want motes to have, should maximize throughput and/or minimize collisions.

- C The set of channels (radio frequencies) available for communication. Members of C are represented by consecutive integers starting with 1.

  - c Some channel from the set of channels C where $1 \leq n \leq |C|$.

  - c_{base} Base channel, the default channel for communication and the lowest numbered channel available.

- β Channel capacity, the bandwidth capacity of a single channel.

- γ Capacity factor, the number of channels with capacity β required to satisfy demand.

- φ Redundancy factor, the number of logical redundant networks (LNETs) to divide a motescape into.

1.4 Research Goals and Motivation

The goal of our research is to precipitate the use of redundant cyber physical systems resulting in a more secure and reliable system.

This goal is motivated by foundational needs that have been identified as barriers to CPS adoption by a group of leading CPS researchers. These needs include insufficient cyber-security, inadequate wireless communication, and a fundamental lack of scalability[8]. Our motivation for using redundancy to address these needs stems from

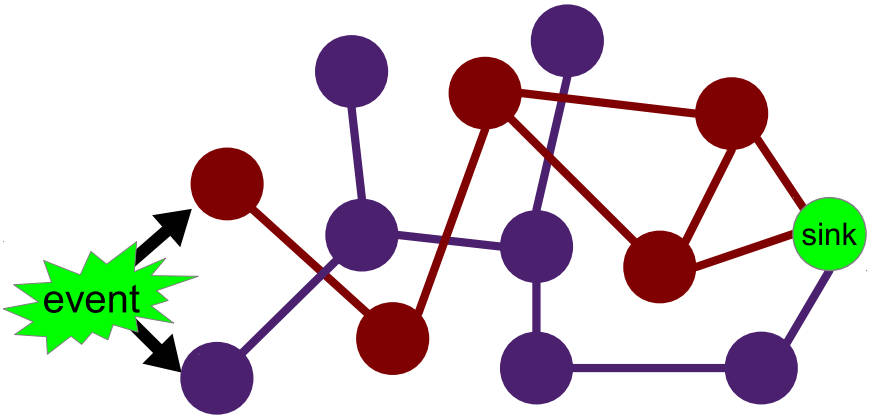


Figure 1.3: Two networks communicating an event redundantly to a sink.

the expectation that it will become feasible to have redundancy as a standard feature. This requires that mote hardware continues existing trends towards extremely low per-unit costs and minuscule form factor[1, 15, 16]. Given this we assume the existence of dense mote populations, whether dictated by application requirements, deployment circumstances, or as an intentional design measure. We then leverage this property of dense populations to address outstanding issues within the field of CPS as enumerated in the concept map shown in figure 1.2. Table 1.4 maps the relationship between these concepts and our research, how we address these concepts is then described in greater detail below.

1.5 Issues identified and how they are addressed

In this section we will provide a road map of our research. We identify the issues addressed by our research, define the issue, explain the relevance of each issue, and describe how we address each issue giving links to relevant chapters and illustrations.

Concept	Metric of relevance
Applicability to CPS (1.5.1) The applicability of the protocol to wireless sensing devices in a distributed, networked environment	• Distributed algorithms (ch. 3. Density, 4. Partitioning, 5. Protocol) • Consideration of network data propagation (sec. 5.2.1. Micro state and macro state) • Use of wireless network simulation models (sec. 6.2. Simulator models)
Scalability (1.5.2) The ability to tolerate fluctuation in demand for bandwidth caused by changes in population density	• Dynamic channel allocation in response to changes in mote population density (ch. 5. Protocol, 6. Simulation) • Scale bandwidth availability in the face of rising demand (ch. 3. Density, 6. Simulation) • Reduction in failed transmission attempts (ch. 3. Density, 6. Simulation)
Cyber security (1.5.3) "The ability of a system to continue operating satisfactorily when stressed by unexpected inputs, subsystem failures, or environmental conditions or inputs that are outside the specified operating range." [8]	• Improvement in availability (3. Density, 6. Simulation) • Facilitate a means to detect malicious data (section 1.5.3)

Figure 1.4: How concepts from figure 1.2 are addressed within this dissertation

1.5.1 Applicability to cyber physical systems

The notion of cyber physical systems is fundamental to our research. All protocols and algorithms given in this dissertation assume distributed wireless communication and our simulations use models apt for CPS.

1.5.2 Scalability

In chapter 5 we give a protocol that provides automatic network capacity scaling by dividing mote populations into multiple logical networks (LNETS) or combining multiple LNETS in response to fluctuating population density as estimated using methods from chapter 3. Each LNET is allocated its own communication frequency, dividing collision domains, reducing resource contention and increasing available bandwidth[17]. The result is the ability for network capacity to scale up or down in response to demand addressing the requirement for scalable methodologies given in figure 1.1.

1.5.3 Cyber security

Cyber security is a broad topic applicable to nearly every aspect of computing. Two areas that are identified in figure 1.1 are *resilience* and *intrusion detection*. Resilience refers to the ability of a network to function under adverse conditions and to recover from failure or attack. This is particularly important in a CPS due to its defining characteristic of autonomy and the criticality of application[5, 9].

We enhance cyber security first by increasing availability which along with confidentiality and integrity make the three fundamental properties of information security[18]. We bolster both **system availability**, the ability of the CPS to operate, and **data availability**, the odds that a packet will be successfully transmitted

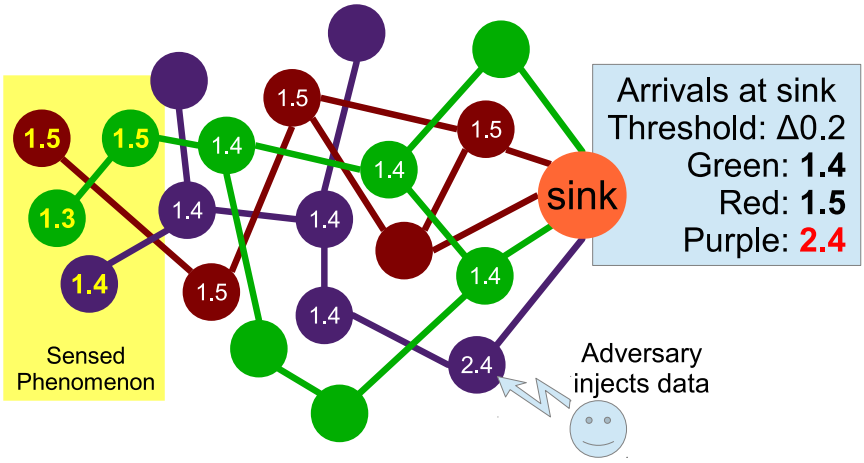


Figure 1.5: Use of redundancy to detect malicious data at sink

from source to sink. System availability is improved because each partition created (4. Partitioning, 5. Protocol) serves as a redundant source of event reporting. This allows the CPS to tolerate the loss of up to $\varphi - 1$ channels of communication, where φ is the number of partitions or *redundancy factor*. Data availability is higher because the probability that at least one report of a given event is delivered increases by up to $\varphi - 1$ orders of magnitude.

Furthermore, the creation of redundant partitions that function autonomously constrains any compromise of a single mote so that it does not affect operation within the other partitions. And finally by providing redundant reports of events, a potential means of intrusion detection is created. The ability of a data sink to compare multiple accounts of each event may reveal malicious data, providing evidence of intrusion as well as a means of mitigating the effect of the malicious data.

By augmenting availability and providing a potential means of intrusion detection we addresses a subset of the cyber security concepts described in the concept map show in figure 1.1.

1.6 Contribution

Our contribution to the field of CPS are the LNET operational protocol, and two algorithms we developed in order to implement it.

The LNET protocol (see 5. Protocol) is used by motes to make use of the techniques and provide the benefits described in this chapter. In the process of implementing this protocol we developed the turn-taking algorithm (see 4.2.5) for partitioning motes into *even* and *overlapping* partitions (see 4. Partitioning). And in order to determining the number of LNETs to partition motes into we developed a means to estimate population density in a physical distributed environment that mitigates the error introduced by edge motes (see 3. Density).

1.7 Organization

The simplified goal of our protocol is to automatically partition a population of motes M to provide *maximum redundancy* and respond to demands for *bandwidth*. To accomplish this we need to be able to derive population density which we accomplish in chapter 3, divide motes into effective partitions which we do in chapter 4 and finally we need a protocol that drives the process which we give in chapter 5.

In the remaining pages we discuss topics relevant to our work and provide details of our methodology, algorithms and findings. We first give a review of literature relating to this work and then proceed in a bottom up fashion explaining the intricacies of calculating density, the problem of edge effect, and how motes can self partition. Then in chapter 5 we present the network protocol and state diagram that make use of density calculation and partitioning to address the problems identified in this

chapter. Finally we present our simulation parameters and results. Table 1.6 gives a brief description of the chapters in this dissertation.

Chapter	**Topic**
1. Introduction	Presents an overview of the problems we address, a description of our work and how it addresses the identified problems.
2. Literature Review	Discusses literature relevant to our work.
3. Density	Defines density and discusses methods for calculating it as well as some of its associated pitfalls. Presents a means of estimating density that mitigates edge effect.
4. Partitioning	Discusses methods for and issues with partitioning. Presents a method for dividing motes into evenly distributed overlapping partitions.
5. Protocol	Presents our LNET protocol which uses the density and partitioning algorithms above to address the problems previously identified in this chapter.
6. Simulation	Presents the assumptions and models used in our simulations, and the data resulting from our experiments.
7. Conclusion	Presents our conclusions, discuss additional research opportunities.

Figure 1.6: Chapter descriptions

Chapter 2

Literature Review

In this dissertation we give a method for mitigating the negative effects of highly dense networks and improving the general resilience of wireless communications in cyber physical systems. In this chapter we present literature that is relevant to the problem of network reliability, network contention and information security as related to cyber physical systems. As the field of CPS is relatively new much of the applicable research is related in a more general or tangential manner.

2.1 Wireless communication

The ability of devices to communicate effectively and efficiently in wireless networks is the heart of the problem we intend to address.

In 1987 Boris Pittel published the widely referenced *On Spreading a Rumor* [19] which laid the groundwork for determining probabilistic bounds on disseminating information through an ad-hoc wireless network with no central control point. Pittel characterizes the problem as a situation in which a node wishes to spread a rumor to all other nodes. At each stage of rumor spreading, each *knower* randomly chooses a

single neighbor to tell the rumor to. The choice of neighbor is independent of previous choices or whether it is a knower or an ignorant. The author then derives probabilistic equations for determining how many knowers exist after t stages of rumor spreading and how many stages are required before all nodes have heard the rumor. We use Pittel's equations to determine boundaries on waiting periods whenever our protocol must disseminate information.

The article *Reliable Broadcasting in Random Networks and the Effect of Density*[20] builds on the Pittel paper and *The shortest-path problem for graphs with random arc-lengths* by Frieze and Grimmett[21] to show that by using the *push protocol* to transmit data to random neighbors, the speed with which data saturates the network is essentially unaffected by the fact that most links are missing and speeds are nearly equivalent to that of a fully connected network. Although the authors examine density and its effect on performance, they concentrate on large random networks whose nodes have small average degree and their computations primarily are concerned with mitigating *sparse* networks rather than the effect of high density. The simulations appear to be more calculation based rather than attempting to model physical processes, for instance rather than detecting edges they simply use a formula to specify likelihood of being an edge node based on density. This would be problematic for our work since we intend to show methods to mitigate the effect of edge densities in a variety of field sizes and shapes that would not be amenable to mathematical modeling.

2.2 Density

Density is a simple concept on its face, but there are many subtle sub topics that can render calculation of density and understanding its effect on network performance

difficult. First of all the term density itself must be qualified as it can refer to multiple characteristics. Network density (φ) refers to the number of motes per unit area in an ad-hoc wireless network. Mote density, or degree, refers to the number of neighbors experienced by a given mote in the network. Density is related to robustness since in order to make sound decisions a system must have sufficient data and sufficient data means making sufficient observations of an event, which is directly related to the number of sensors available[22].

Given some basic variables we can easily calculate or estimate network density for a rectangular or circular uniform random mote deployment. However the manner of deployment, whether random or grid based and the degree of uniformity can have a huge impact on the calculation and even meaning of density[23, 24, 22]. Moreover the variables needed to calculate density, may involve a tremendous amount of overhead to calculate and distribute in a distributed ad-hoc environment and may be unreliable[19, 20, 21].

Even if density is known, there is no standard method for determining the *ideal density* which varies depending on a litany of variables such as hardware and application requirements and distribution. This is complicated further when one considers the ability to dynamically modify transmission power and hence the mean degree[25, 26].

A good introduction to the topic of density is the paper *On Wireless Density*[26] by Christian Bettstetter. Bettstetter expands on strong mathematical results from Cheng and Robertazzi[27] and Wu and Li[28] in order to give a comprehensive set of equations and proofs for relating the radio range, field size and degree. We reference these equations for density calculation in section 3.

One of the foundational papers Bettstetter uses is *Optimum Transmission Radii for Packet Radio Networks* by Kleinrock and Silvester[25]. The authors analyze the trade-off between greater connectivity and increased contention. Increasing power to

the radio will widen the transmission range for wireless motes thereby reducing the required number of hops for transmitting data between any two nodes. However this also means a higher degree of connectivity and increased interference (contention). The authors find that there is a transmission radius that optimizes network capacity achieving a throughput proportional to the square root of the number of nodes in the network. The authors also give a formula for calculating the expected degree of a node when nodes are not required to have the same transmission range.

In the article *Impact of Node Density on Throughput and Delay Scaling in Multi-Hop Wireless Networks*, the authors derive formulas based on the work from Kleinrock and Silvester describing how density effects throughput within a system [29, 25]. The authors also discuss issues with the widely used *cell partitioned network model* with regards to density calculations.

2.2.1 Highly dense wireless networks

Existing literature dealing with the effects of density in ad-hoc wireless networks invariably consider data from the viewpoint of mitigating sparseness and neglect the impact of high density. This is understandable given that sparseness has been a more realistic problem in practical usage.

Kuo et. al. prove that higher node density results in a lower required hop count when connecting a source and destination over multiple hops, leading to exponential improvement in throughput. However the impressive claim of exponential improvement in throughput is misleading. Their data shows that as the average degree increases the improvement in throughput very quickly approaches an asymptotic boundary. The results given hinge on beginning with an extreme minimal mean degree of 1 and do not consider the effect of data loss due to insufficient network capacity[29].

Mansouri et. al. state that if their density calculation yields a value greater than 200%, they simply set the density to 200% in order to "avoid an overweight estimation of density"[24].

Bettstetter derives probabilistic bounds to avoid *isolated nodes* in homogeneous ad-hoc networks but does not consider bounds for paralysis due to contention[26].

2.2.2 Probabilistic connectivity

Devices that use radio communications can typically control the power used for transmission dynamically. Higher powered transmissions will travel further and be seen by more recipients resulting in higher average degree. Since power is one of the most critical resources in the motes that make up these systems however, this decision can not be undertaken lightly[30, 9]. Additionally, the range of any mote sensors must be taken into account to prevent adjustments in transmission range from resulting in untenable sensor coverage[31].

The Kleinrock paper gives equations for adjusting transmission range in order to achieve probabilistic connectivity of a desired degree. This result is expanded on in the papers by Bettstetter and Gupta and we reference these equations for ensuring connectivity within bounds of a given probability.[25, 26, 23]

2.2.3 Edge effect

Edge effect, the impact of border nodes on calculations of density is often mentioned in regards to its impact on simulation, but is rarely studied as a subject in its own right. The intensity of edge effect may be quantified by the ratio of *edge motes* to *interior motes*. The contribution of edge nodes when measuring mean degree results

in a skewed calculation. This is because the edge nodes must have a lower density than interior nodes despite having the same connectivity properties[26, 27, 28, 25, 29].

A number of authors discuss the problem of *edge effect* in the context of simulation efficiency this is because one of the methods of mitigating the problem is to exclude edge nodes from contributing to sample populations which necessitates simulation of additional nodes in order to achieve a desired sample size. Other recommendations for mitigation are treating the network as a torus in that edge nodes on one side simply wrap to the other side of the map.

Our work requires a protocol for calculating density in a physical system. In this environment edge effect will be a real factor and likely be more pronounced than in a simulation since a physical distribution is more likely to have fjords at the edges (ragged borders) or interior *holes* where no motes exist.

In *On the minimum node degree and connectivity of a wireless multihop network* the authors relate range, size and degree but do not draw conclusions specific to any particular application. The author's treatment of edge effect assumes a simulation environment and makes no provision for physical hardware or the necessity of a distributed algorithm for calculation and dissemination. Applying these ideas to a physical system presents a more concrete problem since there may be no way to determine edge nodes, and edges may be complex due to physical obstructions.

The article *Critical Sensor Density for Partial Coverage Under Border Effects in Wireless Sensor Networks*[23] is one of the more thorough reports on border effects. Drawing heavily on previous work by Lazos and Poovendran[22] the authors determine the requirements for calculating edge effect for motes deployed into irregular shaped fields and show the significance of border effects within these fields taking into account both density and range. Previous works had neglected to consider sensing (or radio) range which can play a critical role in manipulating density and coverage.

This work illustrates the rapidly increasing complexity of determining edges in a wireless network whose motes occupy a complex layout. While possible and useful in simulation, it quickly becomes prohibitively complex for real world application, especially for less regular shapes, and would require specialized hardware to determine spatial positioning.

The authors admit to making unrealistic assumptions regarding the geometry of their layouts and recommend approximating irregular motescapes by circumscribing similar regular shapes which is only useful when we have foreknowledge of the shape. In section 3 we will examine border, or "edge" effect and attempt to mitigate it *regardless of shape* in a simple, practical and unassuming manner.

2.3 Coverage and Partitioning

Our proposal involves partitioning a dense network into redundant less dense networks each of which provides sufficient coverage on its own.

The concept of using overlapping coverage for the purposes of redundancy does not appear frequently in literature. In *Integrated Coverage and Connectivity in Wireless Sensor Networks*, Wang et al. mention fusion based distributed detection methods as a means to reduce false data acceptance via multiple coverage, though these methods are done within a single homogeneous network[32].

The process of dividing a network is referred to as *partitioning* which is a well studied branch of graph theory. Coverage and partitioning are related since partitioning a network will result in multiple networks, each with a different degree of coverage than the original.

Calculating coverage is one of the fundamental problems with wireless ad-hoc networks[33]. Coverage may be defined as a performance metric for quantifying

how well an area is monitored by the sensor deployment. Lazos and Poovendran pose the coverage problem as a set intersection problem. This allows the authors to analyze layouts that go beyond the simple shapes that are used in the vast majority of literature on this topic[22].

Methods of using multiple radio channels to alleviate congestion and improve bandwidth are discussed in [34]. In this paper channel assignment methods are categorized as **fixed**, **semi-dynamic** and **dynamic**. We use a hybrid of fixed channel and dynamic channel assignment that is most similar to *Component Based Channel Allocation* (CBCA) introduced in [35]. CBCA allocates channels based on a necessary data flow, however we emphasize allocation that results in even distributed coverage.

Component level channel assignment is the least complex method of channel sharing and according to [33] has the practical advantages of using COTS hardware, no synchronization requirements or channel scheduling overhead and only prudently switches channels.

Two variations of the problem of determining k-coverage of sensors in a distributed network is investigated in [36]. The variations of the problem are that of *unit disk* coverage and *non-unit disk* coverage.

Once network density has been determined and the redundancy factor has been computed we must partition our motes into subnetworks.

2.3.1 Partitioning methods

The simplest method for partitioning a dense graph is for each node to choose randomly from a set of partitions. This may be adequate for the needs of a given application, and is likely an improvement over exact grid based placement[37]. With random selection, if our network of randomly distributed motes contains twice the

population necessary for adequate coverage of its area, and each mote then randomly selects one of two networks to join, the result will be two random networks, each capable of adequate coverage.

While random selection is viable, we describe an algorithm that produces better partitions in chapter 4.

Singh et al. expand on a seminal article from Barnes and Hut giving a way of converting a system of nodes into a regular grid called a *particle mesh* so that it can be manipulated more easily. This involves superimposing a grid over the nodes in the graph and then translating the nodes to the nearest grid point. The accuracy of the final solution depends on how fine the chosen grid is[38].

Although this is an elegant partitioning method, the hardware comprising a motescape is unlikely to have access to the necessary spatial data for implementing this.

2.4 Deployment

We do not delve deeply into alternate methods of deployment, but assume uniform random distribution of motes throughout a given area. There has been some research into the implications of various methods of deployment however. Most commonly the differences between manual grid based deployment and random deployment.

In an article titled *Factors that may influence the performance of wireless sensor networks*[24], the authors discuss uneven distribution and describe physical mote distribution methods such as column or grid based and the effects these have on the space between motes. Zhang and Hou examine the implications of grid based deployment concluding that contrary to intuition, grid deployments tend to render asymptotically lower node density than random deployment[37]. This lends credence to our decision

to use randomly distributed motes for our simulations though this decision was initially predicated on a desire to avoid unnecessary assumptions, embrace simplicity whenever possible and to allow flexibility in the application of our protocol.

2.5 Multi-channel

There has been a good deal of research into multi-channel communication schemes, but existing approaches address issues such as optimization, Quality of Service (QoS) and reliability within a single homogeneous network [39, 40, 24, 41, 42, 29]. These approaches tend to involve complex scheduling and coordination or multiple radios per device which is at odds with motes expected to be extremely resource limited. Moreover existing literature that considers density is nearly universally concerned with sparse networks and do not consider the possibility of highly dense populations.

A paper by Raniwalla et al. titled *Centralized Channel Assignment and Routing Algorithms for Multi-Channel Wireless Mesh Networks*[17] examines converging communications from multiple channels in 802.11 networks using full featured computers with multiple wireless communications cards and centralized algorithms for deriving channel assignments.

The authors assert that multiple non-overlapping frequency channels can be used simultaneously to increase the aggregate bandwidth available to end-users, and that this is common in infrastructure based networks but is "rarely used in the context of multi-hop 802.11-based LANs that operate in the ad hoc mode". The authors also claim that by equipping motes with two network interfaces operating on different channels they can increase the total throughput by a factor of up to 8 over conventional single-channel ad hoc network architecture.

These results are interesting in that they show that leveraging multiple channels of communication can be an effective technique for improving wireless communication in ad-hoc networks. However the purpose of this paper is strictly improvement to throughput and neither network density nor its effect are addressed. Furthermore the authors' requirement of two radios per mote impose hardware requirements that we wish to avoid.

Architecture and evaluation of an unplanned 802.11b mesh network by Bicket et. al. [43] is an examination of the effects insufficient density has on multi-hop mesh networks. This paper covers many topics that mirror our proposed research, the authors present results of multihop communication and examine the effects of density on connectivity and robustness of communication. The results are presented only from the perspective of achieving *sufficient* density. The possibility of overly dense networks is never discussed, and simulation data is only given up to the point of sufficiency.

2.6 Simulation models

Piyush Gupta and P. R. Kumar authored *The Capacity of Wireless Networks*[44] which among other contributions defines *the protocol model* and *the interference model* which are widely followed for determining success and failure for transmissions in a multi-channel network[45, 46, 47, 34, 48, 49]. These models state that a transmission from m_i to m_j, is successful only if all devices that could interfere are silent during the entirety of the transmission. This allows for the *hidden terminal problem* whereby a node inadvertently interferes with a neighbor's reception because it is too far from the transmitter to see its transmission.

The predominant simulation model used for density studies is the *cell partitioned network model*. This model is popular for its simplicity and ability to render statistics in a user friendly manner[29, 50].

The cell partitioned model involves grouping multiple motes into a cluster, then all motes in a given cluster only communicate within the cluster. Any communication that needs to go outside of the cluster must be relayed through some chosen router node. This model then restricts communication to one transmission per cell per timeslot in order to avoid interference[51].

This model is useful for simulation but not considered practical. The required assumptions of the cell partitioned model can render it unable to reflect the significance of node density on network performance[29, 50].

Chapter 3

Density

3.1 Introduction

The ultimate goal of this dissertation is to provide a means of partitioning dense mote populations in a Cyber Physical System (CPS) to form multiple redundant systems which we refer to as logical networks (LNETs). Before we can effectively partition a single population of motes into multiple LNETs, we must first decide how many LNETs to create. This requires us to know the *density* of our mote population and be able to evaluate it with respect to some *ideal density*. It is very possible that motes in a CPS will not have access to the information required for direct density calculation, and in that case we can instead measure the mean mote degree giving the average number of neighbors for each mote. This gives a value that should be a close approximation of density, however in practice a subtle issue arises in the form of a property called the "edge effect".

Edge effect refers to the fact that the inclusion of motes that lie near the perimeter of a motescape contribute lower degree counts to the mean causing the measured

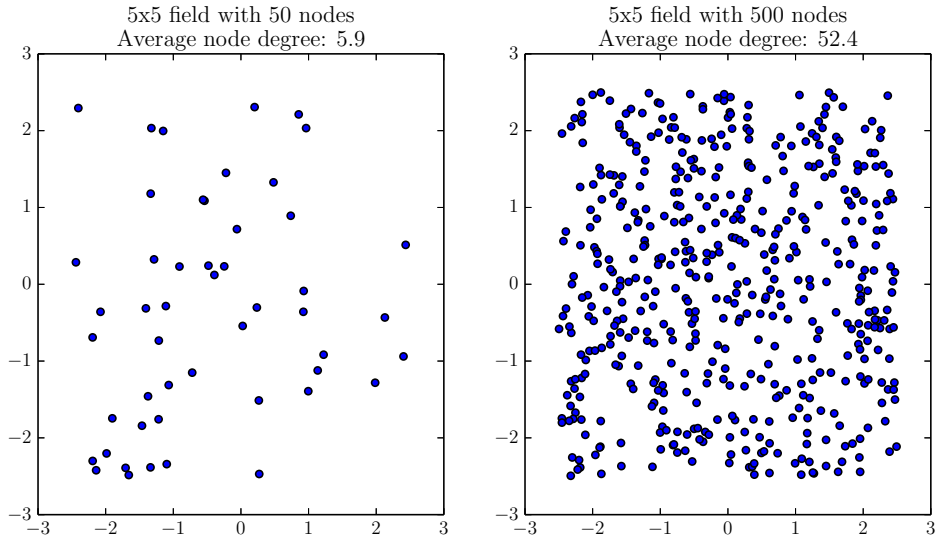


Figure 3.1: Motescapes with varying densities.

mean to be significantly lower than the direct density calculation. To get an accurate density estimate we must find a way to counteract this edge effect.

3.1.1 Motivation

In chapter 1 we describe how knowledge of population density is required to partition motes, and how mote partitioning addresses needs associated with scalability and cyber security (1.1. Identified needs and requirements) by responding to fluctuation in bandwidth demand (1.5.2. Scalability) and improving availability of systems and data (1.5.3. Cyber security).

3.1.2 Definitions

The term density refers to a measure of how many motes exist per some *unit of area*. In chapter 1 we define symbols for both calculated density δ, and measured density μ, as well as the unit of area U, figure 3.1 illustrates an area with different densities.

Degree: $deg(m)$

Degree is a function that gives number of one hop neighbors for mote m and is analogous to the term degree as used in graph theory to denote the number of edges of a given vertex. Given that the number of neighbors a mote has is directly related to its range of communication, a mote's degree can be used as a sample of population size for unit of area U_r, which is the reachable area of m's radio.

Unit of area: U

Unit of area U refers to the size of the unit we use for measuring population density, that is we want to determine the average number of motes in an arbitrary area the size of U. By using a unit of area equal to 1, $U = U_1$ we get a population density with respect to a fundamental unit such as square meter.

Our ultimate purpose for calculating density is to determine how many partitions we can divide a motescape into such that each approaches an ideal degree ℓ. In section 3.1.2 we define degree a mote's neighbor count, and so we will generally define our unit of area to be the area covered by the wireless radio used for mote-to-mote communication $U = U_r$. Though we do no address domain specific sensor issues, one could easily substitute sensor coverage area $U = U_s$, or any other circular area.

Calculated density: δ

Calculated density refers to dividing the area of a motescape into standard units and then finding the expected population of an arbitrarily chosen unit of area within the motescape. Calculated density is directly computed as opposed to measuring populations or taking samples as described in section 3.1.2. Calculated density gives an exact representation with no significant effort, however it requires knowledge of

the area of a motescape A which can be difficult or impossible to calculate depending on installed mote hardware and the physical layout of the motescape. Mote density is calculated with equation (3.1) described by Bettstetter in [26], here the motescape population size $|M|$ is divided by the area A. This is the case where the unit of area is equal to 1 and gives the number of motes we should expect to find on average within any arbitrary unit of area such as one square meter. By equation (3.2) we can multiply the physical density δ_1 by a unit of area U_x in order to obtain the density with respect to that unit of area, this gives the mote population expected within an arbitrary unit of area the size of U.

$$\delta_1 = \delta(U_1) = \frac{|M|}{A} \qquad \text{Physical density} \tag{3.1}$$

$$\delta(U_x) = \delta_1 U_x = \frac{|M|}{A} U_x \qquad \text{Density with respect to } U_x \tag{3.2}$$

Example A 10x10 meter area containing one mote would yield $\delta_1 = \frac{1}{100m^2} = 0.01$, one hundredth of a mote per square meter, or one mote per $100m^2$. If the same motescape had 200 occupants, we would get $\delta_1 = \frac{200}{100m^2} = 2$, or two motes per square meter. To apply a unit of area, assume $U = \pi$ resulting in $\delta = 2 \cdot \pi \approx 6.28$.

Measured density: μ

Density can be easily calculated in a simulation environment when its components are known, however within a distributed network it is likely that density must be measured since there is no inherent access to the required data, and may be no means to acquire it. Measured density is represented by the symbol μ, and is simply the *mean degree* observed within a population of motes. Like computed density, measured density represents the population size we expect to find in an arbitrary unit of area

of a motescape, however there are key differences: *a*) the unit of area is inherent to the system being measured rather than being given; *b*) μ is the mean mote degree, however degree is the number of neighbors observed by some mote m and does not include m itself, hence μ will be one less than δ. Whenever comparing μ to δ, μ is acting as an approximation to population size per unit of area, so we must either add 1 to μ or subtract 1 from δ.

The equation for measuring the density of a mote population as given in [26] is in (3.3), this simply sums degrees for all motes in M and divides the result by the population size. For the sake of efficiency sampling a subset of M would likely be used in a real world environment.

$$\mu(M) = \frac{1}{|M|} \sum_{m \in M} deg(m) \tag{3.3}$$

When compared to calculating δ as explained in 3.2, determining μ requires a large expenditure of resources since, the degrees of a statistically significant number of motes must be measured. Furthermore δ gives a precise value while μ will contain error. The error in μ is due to statistical error margins associated with sampling, and the fact that the needed measurements take some time to complete and so motes sampled early, may change their degree before the process completes resulting in inaccurate measurements, but a larger source of error is that introduced into measurements by the presence of "edge motes" as detailed in section 3.4.

3.1.3 Metrics

To determine whether our density estimation method is effective we identify an objective function in section 3.2 which represents an ideal value we can use for comparison.

The objective function is a calculation of the expected mote density, since it is a calculated value as opposed to a measurement, the impact of edge motes is eliminated and the *true* mean degree being sought is represented. The baseline value upon which we wish to improve is the measured mean degree of all motes in the motescape, so edge effect is not mitigated in any way.

3.1.4 Contribution

Our contribution is an efficient method for estimating the mean degree of a physical population of motes. Moreover our method reduces the impact of the edge effect which affects a simple average and works with complex field shapes. We present simulation results that validate our claim.

3.2 Measuring density and redundancy

The ability to calculate the mean mote density is a basic requirement of our LNET scheme. Simple mean calculation is a straightforward procedure whereby motes determine their own degree and then share this information throughout the network in order to determine the mean. A mote can trivially calculate its own degree which is simply the number of unique neighbors it is able to communicate with directly, that is to say motes within range of radio transmission.

To measure our ability to calculate density we give equation (3.4) as our *objective function*. This is derived from the formula given by Bettstetter in [26] to represent the expected number of nodes for a unit of area. We scale Bettstetter's equation by the radio coverage area U_r which gives density with respect to the transmission coverage of the radio used for wireless communication resulting in a more intuitive value that is directly representative of mote degree.

$$E(\delta) = \delta(U_r) = \frac{|M|}{A} \cdot U_r \tag{3.4}$$

3.2.1 Distributed density measurement

Measuring density, represented by the mean degree of motes within a distributed environment is an instance of the *distributed average consensus* problem[52]. Though this is a simple process using equation (3.3), it is costly in simulation since all motes must locate and count their neighbors and in a distributed environment it requires information exchange between every pair of motes in M which is costly and difficult to manage and given the limited resources of motes it is unlikely they can keep track of which nodes they have data for and which they do not. We wish to avoid this by computing the average degree of all motes using only direct communication between motes (single hop distance).

The iterative equation in 3.5 is widely used for arriving at consensus in a distributed environment[52]. We can use this within each mote in a motescape to iteratively hone in on a shared estimate of some value Δ. In order to calculate the standard deviation we need to know the distribution of degree values, or how many motes have each given degree. This requires that the data being distributed include not only δ, but an array of "bins" in which each element contains two sub elements, element 0 is a degree d_i and element 1 is the number of motes that have that degree $count(d_i)$. Given the mote with the largest degree d_{max} and the mote with the smallest degree d_{min}, the size of $bins$ is at most $d_{max} - d_{min}$.

$$\Delta_m(t+1) = \Delta_m(t) + \frac{1}{|N_m|} \sum_{k \in N_m} (\Delta_k(t) - \Delta_m(t)) \tag{3.5}$$

$$\Delta = [\hat{\delta}, bins] \tag{3.6}$$

$$bins = \{(d_{min}, count(d_{min})), \cdots, (d_{max}, count(d_{max}))\} \tag{3.7}$$

The value Δ in equation (3.6) contains mote m's estimate of network density $\hat{\delta}$ and its knowledge of how many motes have a given degree $bins$, t is an iteration counter, N_m is the set of neighbors for mote m, and $|N_m|$ is mote m's degree. Each mote $m \in M$ initializes Δ_1 using its own degree $\hat{\delta}_m(0) = |N_m|$, and the values in $bins$ are calculated using its own neighbor set. Mote m then broadcasts its values to, and collects estimates from, its neighbors. Once m has collected its neighbor's estimates, it can refine its own by calculating Δ_2 using 3.5, after which it will broadcast its improved estimate. This process repeats until all estimates are within a tolerance of one another at which point motes will have agreed on a network density value $\delta \equiv \hat{\delta}_m \forall m \in M$ as well as the $bins$ which can be used to calculate both the mean and the standard deviation σ. Equation (3.8) illustrates how we can use the contents of $bins$ to determine the mean degree (our density estimate), we will use this in section 3.4 to recalculate $\hat{\delta}$ by multiplying each degree ($b[0]$) by the number of motes having that degree ($b[1]$), then dividing by the number of motes, finally we add one since degree is one less than density due to the fact that the mote observing the degree excludes itself.

$$\hat{\delta} = \frac{\sum_{b \in bins} b[0] \times b[1]}{\sum_{b \in bins} b[1]} + 1 \tag{3.8}$$

Once we have a globally accepted value for δ we can utilize it to determining the number of partitions to create based on need for capacity or desired for redundancy (section 3.3.1).

3.3 Base density and density factors

The notion of partitions based on density implies the existence of a *base value* against which density can be compared, representing one or more of the optimal, minimum or maximum values for a given context. We have thus far discussed concepts associated with density without addressing the scale that determines when a population is dense or not dense, in this section we define two scalar variables each of which require knowledge of scale. These indicators are **redundancy factor** which indicates the level of redundancy as the number of LNETs, and **capacity factor** which determines the total available bandwidth as the number of wireless channels.

3.3.1 Redundancy factor

We measure the network density in order to determine how many partitions to create with the intention of introducing redundancy of communication. Redundancy comes from reporting events via multiple LNETs. The *redundancy factor* (φ) refers to excess coverage of a unit of area. In other words how many times the total motescape population goes over *sufficient population*. We calculate the redundancy factor using equation (3.9) which gives the ratio of density (represented by $\mu + 1$) to ideal density ℓ rounded to the nearest integer. We then limit φ to be no greater than the total number of channels available $|C|$ because φ dictates the number of LNETs we will create, each of which is assigned a unique channel from C. The number of channels

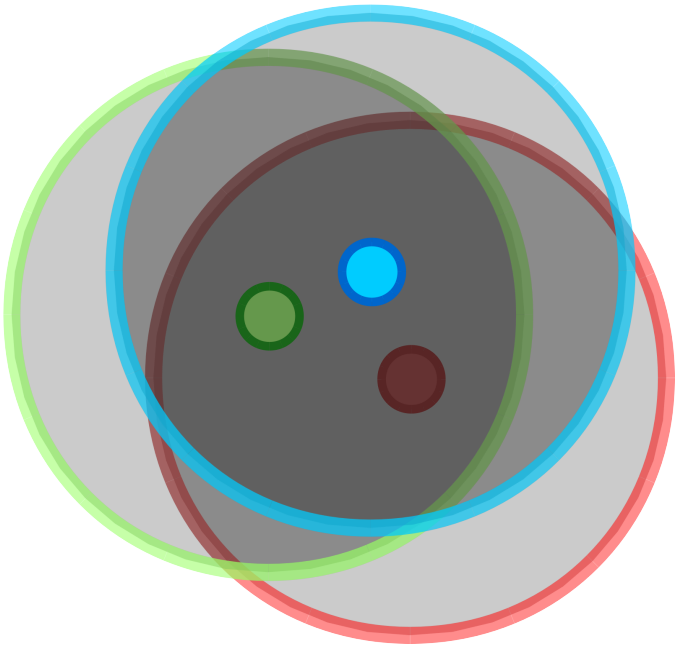


Figure 3.2: Redundant coverage

represented by C is a function of the radio hardware and may vary by geographical region.

$$f = \begin{cases} \lfloor \frac{\mu}{\ell} \rfloor & \text{if remainder} < 0.5 \\ \lceil \frac{\mu}{\ell} \rceil & \text{otherwise} \end{cases}$$

$$\varphi = min(|C|, f) \tag{3.9}$$

Note that although we call φ the redundancy factor, the maximum redundancy is $\varphi - 1$, or one less than the number of partitions since one report of the event is non-redundant.

Figure 3.2 illustrates an event occurring in an area covered by three sensors, one sensor's coverage is required and the others are redundant, so φ is 3 and the event is sensed redundantly twice.

Example Given a motescape with $\ell = 25$, $\mu = 93.75$, $\varphi = round(93.75 \div 25 = 3.75) = 4$, the redundancy factor is 3 because the network *can be* partitioned into 4 networks each having a mean degree of $\mu' \approx \mu \div \varphi$, $\mu' \approx 23.4375$.

Base redundancy value

There is no standard formula for determining an ideal density for a wireless network as it depends on hardware, application and deployment method[24][29]. The goal of finding ideal density should be to obtain coverage while minimizing contention for the shared wireless resource.

In [53] we determine an ideal radio density by simulating multi-hop transmissions through random networks of varying densities in order to find the mean neighbor count (μ) that achieved the best chance of successful data transmission. Our simulations yielded $\mu \approx 17$ and $\sigma \approx 5.2$. However this result is highly dependent on variables such as neighbor activity, number of retries and back-off time.

3.4 The edge effect

In [26] Bettstetter notes that if we were to *measure* the mean degree of a motescape using equation (3.3) we would expect the measured value to closely resemble the objective function in (3.4). We would however find the resulting density to be lower than anticipated due to the *edge effect* which refers to the impact edge motes have on density measurement. An edge mote is a mote within transmission range of the perimeter of the motescape, these motes have fewer neighbors on their 'exterior' side and thus a lower average degree than interior motes. The relative lack of neighbors edge motes have does not impact their communication abilities since the neighbor density available to carry traffic inwards is not affected.

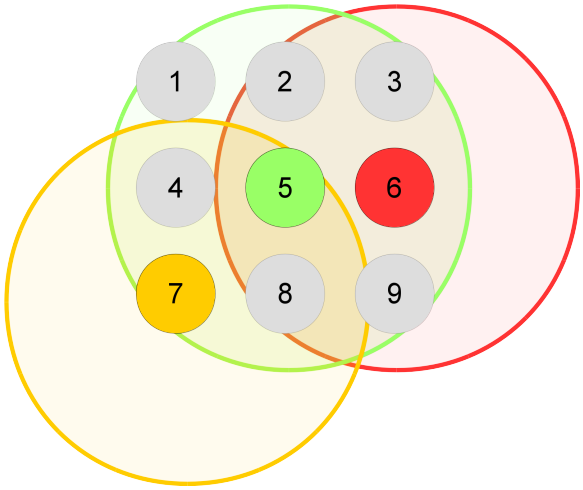


Figure 3.3: The edge effect

The root cause of the edge effect stems from the fact that equation (3.4) is an analytical derivation and assumes an infinite motescape but an actual implementation, either physical or simulated, will have a perimeter.

If we measure average degree using only *interior motes*, those not within transmission distance of the field edge, we will get an accurate representation of expected degree since the motes that would skew our results are excluded from the calculation[25, 29, 26].

Figure 3.3 illustrates the edge effect, having four corners, four sides and a single center node with degrees of 3, 5 and 8 respectively. The measured average degree is $(4 \times 3 + 4 \times 5 + 1 \times 8) \div 9 \approx 4.4$, in actuality all nodes have the same *effective density* as the center node, but have differing degrees since they can only form connections towards the center[26].

Another illustration of edge effect is shown in figure 3.4 in which a central mote positioned near the center, edge and corner choose a set of neighbors that are mutually exclusive from each other.

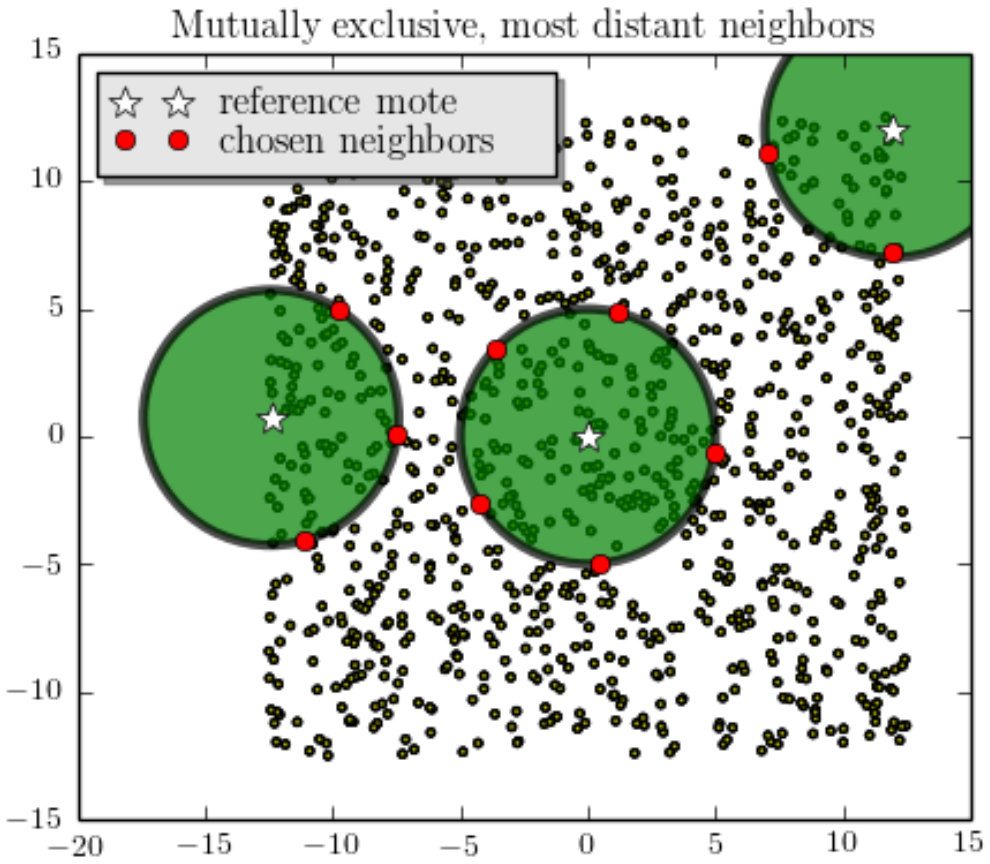


Figure 3.4: Mutually exclusive neighbors of center, edge and corner motes.

3.4.1 Countering edge effect

In section 2.2.3 we describe some simple and effective strategies that are typically used for countering edge effect. Wrapping the edges around to form a virtual sphere or a torus ensures that no edges are ever encountered. Requiring that motes positioned within r_0 of an edge be excluded from the mean calculation ensures that no edge motes contribute to the mean. Both of these methods require knowledge of relative position of motes to the perimeter, moreover if the motescape has a complex or irregular shape or internal discontinuities the computational complexity involved in determining which motes lie on an edge outstrips the benefit of doing so[23]. To our knowledge compensating for edge effect hasn't been addressed for a physical realm.

$$\hat{\delta} = \mu(\{m \in M \mid deg(m) > \mu - \sigma\}) + 1 \tag{3.10}$$

The equation in (3.10) is the logical expression of our approximation. We begin by calculating the mean mote degree μ and its standard deviation σ. This gives us an estimate of density but includes edge motes and is thus skewed. To correct for this we then re-compute density as the mean of motes having degree greater than one standard deviation below the mean, $deg(m) > \mu - \sigma$. We add one to the final estimate because when motes measure degree they do not include themselves but we wish to approximate $E(\delta)$ from equation (3.4) which is a measure of *population* per unit area.

Describing equation (3.10) more explicitly, we first identify the degree values below our threshold of $mu - \sigma$ using equation (3.11), we then calculate our refined density

estimate excluding these values as shown in equation (3.12).

$$bins\prime = \{b \in bins : b[0] > \mu - \sigma\} \tag{3.11}$$

$$\hat{\delta}\prime = \frac{\sum_{b \in bins\prime} b[0] \times b[1]}{\sum_{b \in bins\prime} b[1]} + 1 \tag{3.12}$$

Figure 3.5 provides an illustration of using $\hat{\delta}$ and includes the mean degree of all motes μ, the minimum degree required for inclusion in the calculation of $\hat{\delta}$ $deg(m) \leq \mu - \sigma$, as well as the calculated density we use as our objective $E[\delta]$ and the resulting density calculated with our approximation $\hat{\delta}$. The approximated value is much closer to the objective than the mean, and the visual cue as to the motes excluded from the density calculation make it clear that the majority of motes being excluded are indeed edge motes. Though anecdotal, this example is typical and we show the results of more sophisticated experiments in chapter 6.

Not only does this provide an efficient means for density estimation but it has the advantage that it will work without additional complexity on fields that are odd shaped or contain impediments that would make calculation difficult or impossible to compute otherwise. Figure 3.6 shows a graph with several *holes* representative of natural barriers such as water or buildings. The histogram beneath this figure represents the degree bins used for calculation of the standard deviation.

The idea to use mean and standard deviation as a threshold for inclusion in our degree measurements is based on intuition. The exact threshold of one standard deviation below the mean comes from extensive simulation with various factors of the standard deviation for differing motescape configurations.

To measure the accuracy of our estimated density, we generate random rectangular motescapes with sizes ranging from 300x300 to 2000x2000.

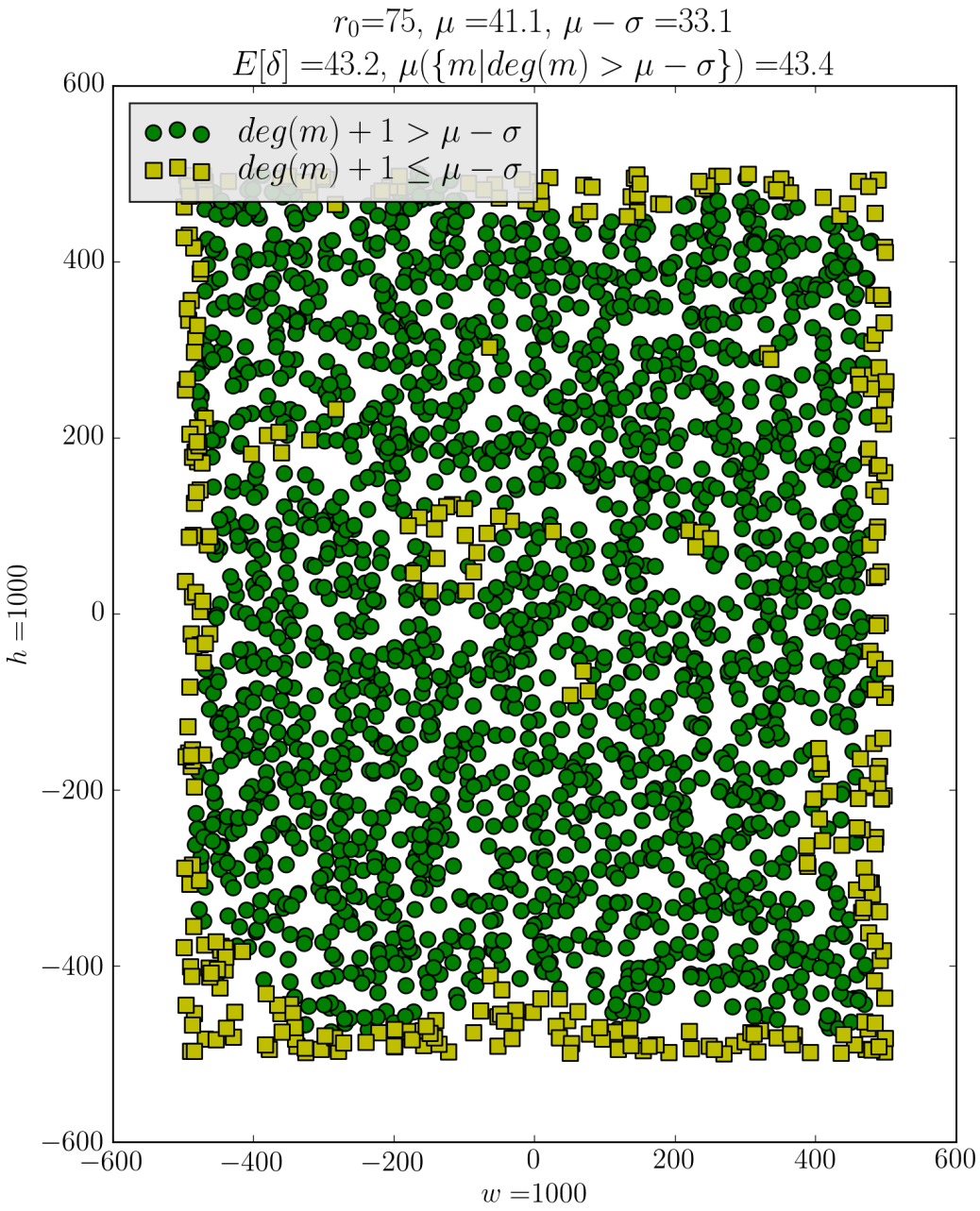


Figure 3.5: Motes excluded from density calculation by statistical estimation.

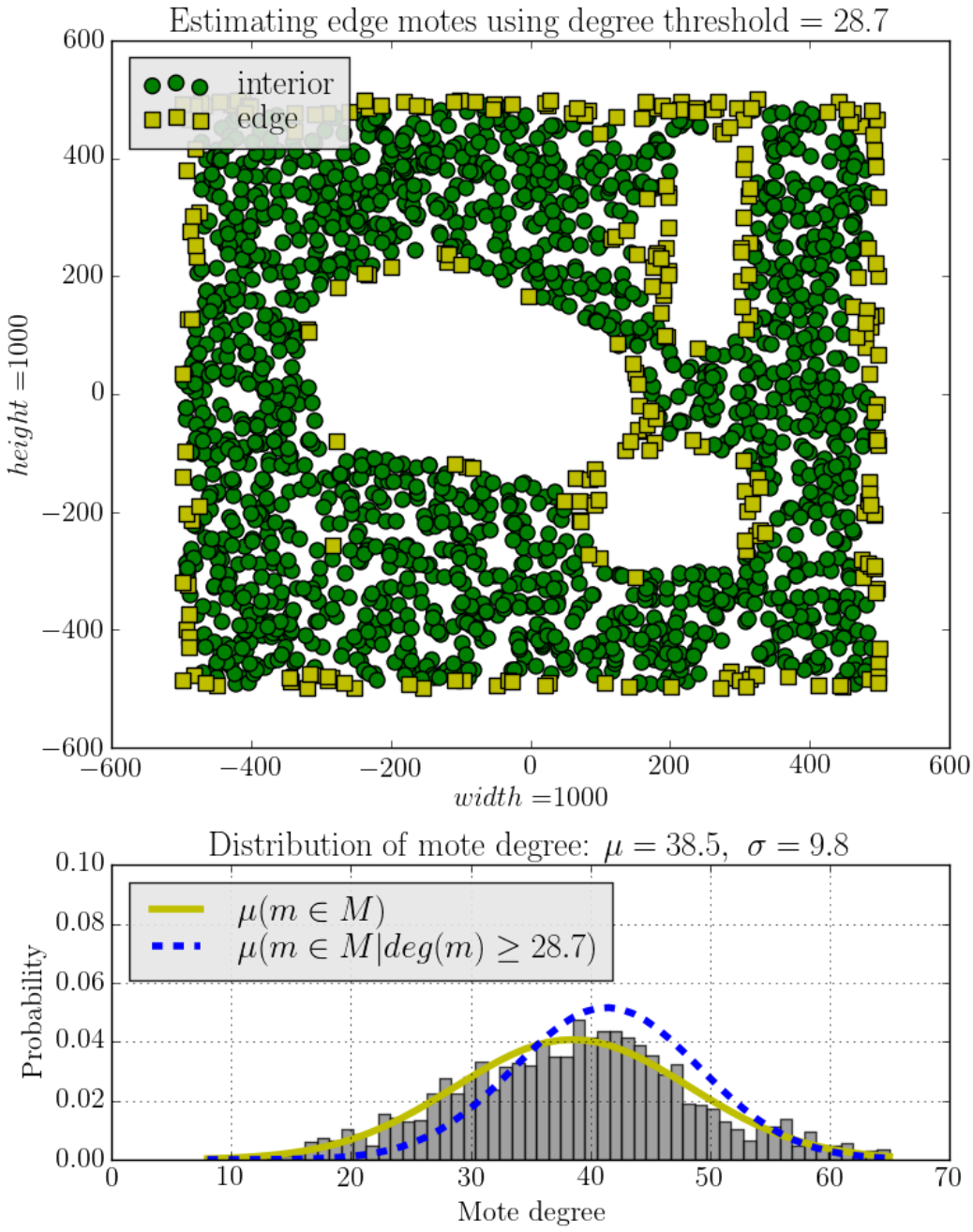


Figure 3.6: Odd shaped network

The size of the mote population is calculated using equation (3.13) with the maximum area $A = 2000^2$, unit of area $U = \pi 75^2$, and $\delta = 6$ is the density we want the motescape to have once populated.

By holding the expected density value $E(\delta)$ invariant, the only variable that affects our density measurements is the size and/or shape of the motescape.

$$population = \delta \frac{A}{U} \tag{3.13}$$

With each experiment conducted we compile the expected density $E(\delta)$ using equation (3.4), the measured density $\mu(M)$ from equation (3.3), and the approximated density $\hat{\delta}$ measured in motes selected by our approximation algorithm from (3.10)). In figure 3.7 we graph the deviation of δ and $\hat{\delta}$ from the objective function. It is clear that as the field size grows the impact of edge nodes decreases. However our approximation provides a more accurate density value than the simple mean approach, even as field size grows very large.

Example: Assume we have a 10x10 meter field ($w = 10, h = 10$) containing 250 motes ($|M| = 250$), and transmission range of a meter and a half ($r_0 = 1.5m$). We can calculate our unit of area ($U_r = \pi r_0^2 \approx 7.07m^2$) and total field area ($A = wh = 100m^2$). Using our objective function in (3.4), we multiply our population size by the ratio of unit coverage area to total area to get the expected density: $E[\delta] = 250\frac{7.07}{100} = 17.675$ which should be the average degree of interior nodes.

This objective function depends on a random distribution and in fact will not be accurate for precise grid aligned nodes where changes in the transmission power will result in either no change to the number of neighbors or the sudden inclusion or

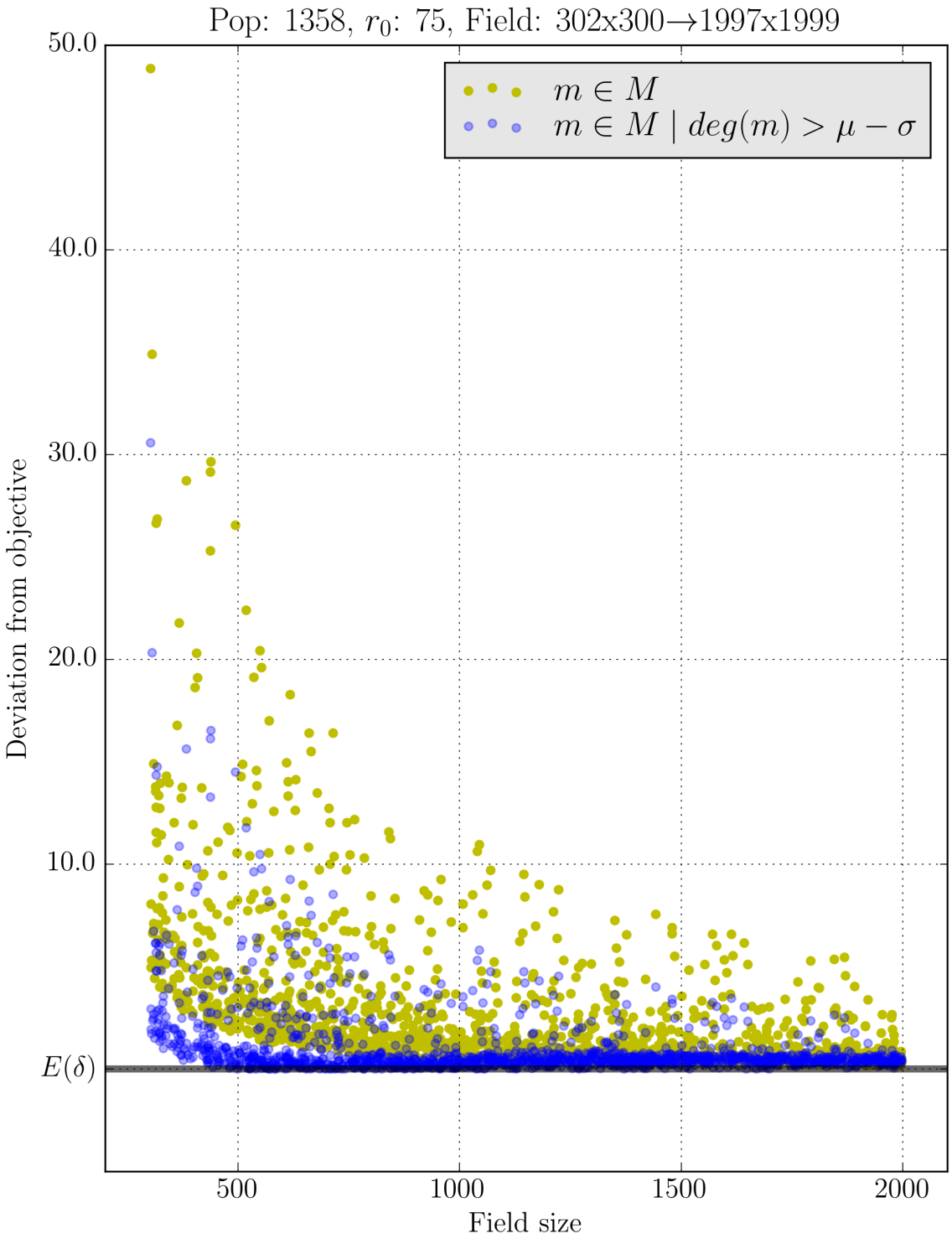


Figure 3.7: Deviation from $E(\delta)$ when varying field size.

exclusion or a large group of neighbors since neighbors on all sides are equidistant. For example if node 5 in figure 3.3 expands or contracts its radio range, even a small amount it can change the degree from 0 to 4, in a large grid this same effect would take place for all nodes in the grid simultaneously if they are all using the same radio radius.

3.4.2 Effective density

In his article on node degree and connectivity Bettstetter gives equations to show the effective area and node count of a two dimensional square field after taking the edge effect into account[26]. We expand Bettstetter's methods to work with any 2d rectangular motescape in (3.14) or 3d cuboid motescape in equations (3.15).

Two dimensional motescape

We can split the total area of a two dimensional rectangular field into edge area (3.14b) and interior area (3.14a) by calculating the area of a rectangle using a width and height that has been reduced such that edges are not included. The width of one edge is the radius of our radio range r_0, we must multiply it by two, once for each side of the motescape to get the formula in (3.14a). We can then compute the amount of edge area as the remainder after subtracting the interior area from the total area using equation (3.14b). Finally with equation (3.14c) we take the ratio of edge to total area to quantify the intensity of the edge effect for a given rectangular field with the requirement that the range of radio transmission can be fully encapsulated within

the perimeter of the field. The interior area A grows faster than A_{edge} so the edge effect becomes less significant as field size grows or r_0 shrinks.

$$A^2 = wh \qquad \text{Area of 2d rectangular motescape}$$

$$w_{edge} = h_{edge} = 2r_0 \qquad \text{An edge is within } r_0 \text{ of perimeter}$$

$$A^2_{interior} = (w - 2r_0)(h - 2r_0) \tag{3.14a}$$

$$A^2_{edge} = A^2 - A^2_{interior} \tag{3.14b}$$

$$intensity = \frac{A^2_{edge}}{A^2} \tag{3.14c}$$

In these equations A^2 is the total area of a two dimensional field, w and h are its width and height, r_0 is the transmission radius of the radio used for communication which determines the width of the field's *edge*. These values allow us to state the ratio of edge area to total area in equation (3.14c) which represents the intensity of the edge effect.

Three dimensional motescape

Both contention due to density as well as the edge effect will be exacerbated by cyber physical systems where the motes are embedded within three dimensional space. This will result in a higher edge to area ratio, hence more nodes will negatively impact density calculation. Li, Pan and Fang study the subject of density in three dimensional networks in detail [47].

In determining the effect of edges within a cuboid area, we proceed in much the same way as with a rectangular one. We subtract twice the size of the edge from each

dimension when calculating the area in equation (3.15a) giving the area of an inner cuboid which does not include the edges. We then use equation (3.15b) and subtract the inner area from the total area to get the area occupied by edge motes. Finally we determine the intensity of edge effect for the given shape with equation (3.15c).

$$A^3 = whl \qquad \text{Area of cuboid motescape}$$

$$A^3_{interior} = (w - 2r_0)(h - 2r_0)(l - 2r_0) \tag{3.15a}$$

$$A^3_{edge} = A^3 - A_{interior} \tag{3.15b}$$

$$intensity = \frac{A^3_{edge}}{A^3} \tag{3.15c}$$

In figure 3.8 we show the relative intensity of edge effect for a cube, square, sphere and circle as the area of the field increases.

3.5 Proactive density manipulation

With modern wireless communication, the radio transmission power is software configurable within a given range, we can utilize this to achieve a desired mean degree within a motescape. Since μ represents the average mote degree, then if we scale the radio coverage area by a factor of k, the expected degree of m will likewise be reduced by a factor of k.

In equation (3.16) we take the ratio of the desired density $\hat{\delta}$ to the measured mean degree μ, ensuring that we adjust by one since degree is from the perspective of some mote and it does not include itself.

$$k = \frac{\hat{\delta}}{\mu + 1} \tag{3.16}$$

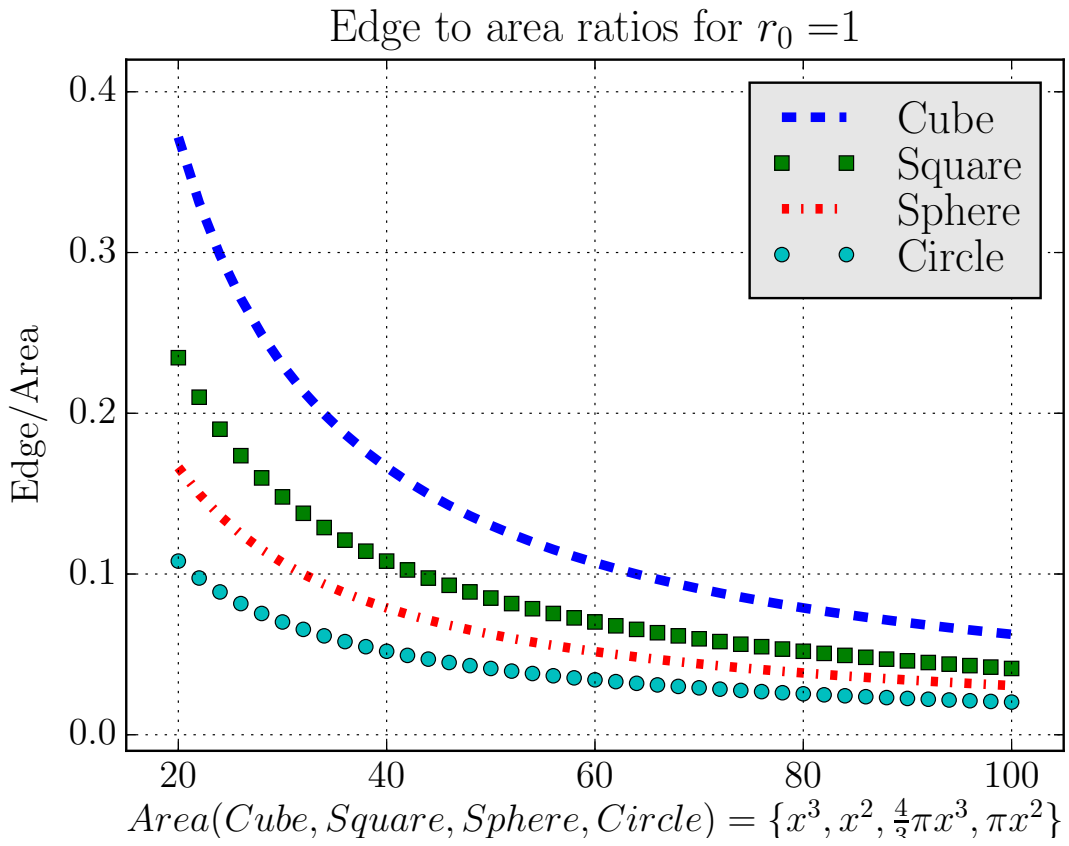


Figure 3.8: Impact of field size on edge

The equations in (3.17) illustrate how we can scale a unit of area U by a percentage k. If r_a is the radius used by the motes in M, and we determine a desired percentage of change using equation (3.16) then we can calculate a new transmission radius r_b that will result in the desired mean degree as $r_b = r_a\sqrt{k}$.

$$\begin{aligned} U =& \pi r_a^2 \text{ rewritten gives } r_a = \sqrt{\frac{U}{\pi}} \\ kU =& \pi r_b^2 \text{ rewritten gives } r_b = \sqrt{\frac{kU}{\pi}} \\ r_b =& \sqrt{k} \cdot \sqrt{\frac{U}{\pi}} = \sqrt{k} \cdot r_a \end{aligned} \tag{3.17}$$

3.6 Conclusion

In this chapter we described the meaning of density and the unit of area for which it is calculated. We explained the need to determine population density and how it fits into the LNET partitioning protocol. We gave a formula for computing population density and reasons why it may not be possible to calculate in a distributed environment. We explained how mean degree is related to density, and gave a method by which distributed motes can arrive at a consensus for mean degree. We then showed how measuring mean degree results in a flawed density estimate due to the inclusion of "edge motes". We derive equations to quantify the effect of edge motes within both a two dimensional and a three dimensional motescape. And finally we give a method for approximating density by using the measured mean degree and eliminating contributions that come from motes that are statistically likely to lie within the edge of the motescape.

Chapter 4

Partitioning

4.1 Introduction

Given a densely populated Cyber Physical System (CPS), we wish to divide its motes into overlapping logical partitions (see figure 4.3), each of which will then be assigned exclusive access to a wireless communication channel. By partitioning motes in this manner we divide collision domains thereby reducing contention for bandwidth which results in improved throughput and reliability[17, 34]. Additionally we gain the ability to provide a degree of redundancy with respect to event sensing and transmission of sensory information to the data sink where it will be consumed, resulting in *a*) an exponential increase in the likelihood that an account of a given event reaches the data sink; *b*) potential for the data sink to compare multiple event reports to ascertain possible compromise or damage within a network and mitigate its impact[53].

4.1.1 Motivation

In 1. Introduction we outline how partitioning dense mote populations into multiple, overlapping logical networks (LNETS) that are more sparsely populated fits into our scheme for addressing needs associated with scalability and cyber security (1.1. Identified needs and requirements). Partitioning into independent LNETs allows us to provide redundant sensor data which improves fault tolerance (1.5.3. Cyber security). And bandwidth scalability (1.5.2. Scalability) is provided since each LNET is allocated its own communication channel.

4.1.2 Definitions

Partition

We define a partition of the motes in M as the family of sets $P = \{P_1, \cdots, P_\varphi\}$, where φ is the redundancy factor as defined in section 4.1.2. P is a partition of M if and only if all of the following conditions hold:

Rule	**Explanation**
1. $\emptyset \notin P$	P is not empty;
2. $\bigcup_{n=1}^{\varphi} P_n = M$	Every mote in M is in P;
3. if $P_a, P_b \in P$ and $P_a \neq P_b$ then $P_a \cap P_b = \emptyset$	Each mote in M is assigned to exactly one partition.

Figure 4.1: Formal definition of partition

Redundancy factor: φ

The redundancy factor φ is described in section 3.3.1, and represents the number of partitions we will create, it is an integer derivation of the mean degree μ divided by the ideal density ℓ.

1. Each partition should cover roughly the same region as the original unpartitioned motescape F, albeit with less resolution;
2. Partition members should be evenly distributed throughout the motescape;
3. Each partition should have approximately the same number of members;
4. Each partition should have approximately the same mean density.

Figure 4.2: Desirable partition attributes

Since our partitions are intended to function as redundant data sources it is imperative that partitions have sufficient population density such that each retains adequate connectivity and coverage with regards to both communication range and sensor exposure. This is expressed by all three of the goals listed above. Clearly partitions can not function as a redundant event sources if they do not each cover the same area in which the event occurs. If our partitioner fails to produce partitions with proportionate spatial distribution and population sizes the result will be unequal connectivity properties leading to disconnected segments, sparse coverage, or sub-optimal operation.

4.1.3 Metrics

To evaluate the effectiveness of a partitioning algorithm we simulate a dense motescape, execute the partitioner and measure the resulting partitions with respect to the properties being tested. The measured properties are compared against expectations such as defined by the objective function given in (3.4).

4.1.4 Contribution

We present the **turn-taking** algorithm in section 4.2.5 by which a set of motes co-operatively partition themselves into evenly distributed, overlapping logical networks (see figure 4.3f for illustration).

4.2 Partitioning approaches

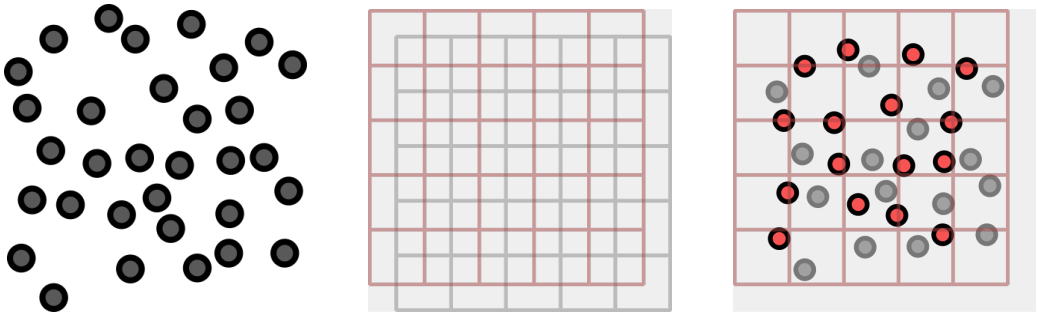
During partitioning we wish to divide motes into evenly distributed, overlapping sets. So if we have a single set of motes that we wish to divide into two LNETs, we would like each of the resulting partitions to contain approximately the same number of motes, and we want the distance between motes in each partition to be even and somewhat "gridlike".

4.2.1 Metric for ranking neighbors

Partitioning requires identification of a property motes can use to select suitable neighbors or reject poor ones. The desirable properties of a partition given in figure 4.2 indicate that proximity between motes is a good candidate for a ranking system. Modern radio hardware used in network communication has access to a property called the Received Signal Strength Indicator (RSSI) for each message received[54, 55]. While not considered suitable for measuring precise distances, RSSI can be used as a relative measurement to distinguish between neighbors. A higher RSSI value indicates a stronger signal and hence a short distance. We use the RSSI as a metric by which a mote can rank its neighbors by relative proximity.

4.2.2 Random

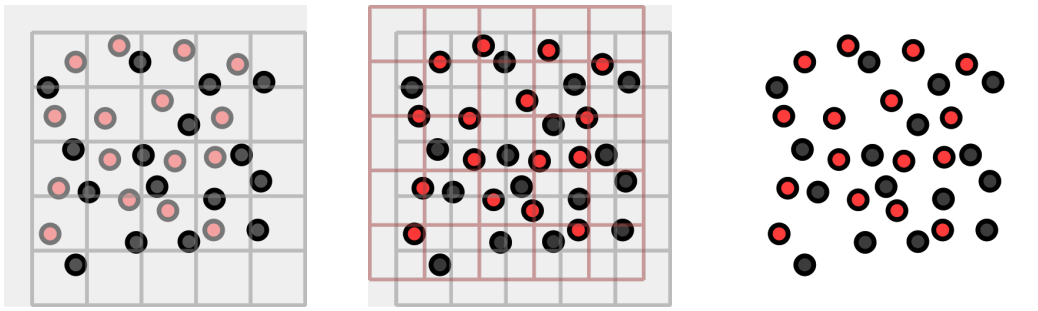
Assigning motes arbitrarily to a partition may provide an acceptable distribution but we strive to improve upon it and to decrease the likelihood of problems due to unlucky choices.



(a) 32 unpartitioned motes

(b) Overlapping 5x5 grids

(c) Motes nearest red intersections



(d) Motes nearest gray intersections

(e) Partitioned motes

(f) Grids removed

Figure 4.3: Creating two overlapping partitions

4.2.3 Offset grid

Figure 4.3 illustrates the use of two identical square grids made up of s blocks per side, the blocks having sides equal to the radius r_i needed to divide the measured density by two, this can be calculated using the formulae described in equation (3.17). The grids are laid atop one another then offset horizontally and vertically by $\frac{1}{2}r_i$. Since the cell size is the ideal distance between motes, we can create partitions for each grid by selecting the mote nearest to the internal intersections.

The grid partitioning shown in figure 4.3 aids in understanding our goal, but it isn't feasible in a distributed environment absent global spatial awareness by all motes. Moreover the number of grid intersections is not arbitrary, and it becomes difficult to position the grids as the number of partitions increases.

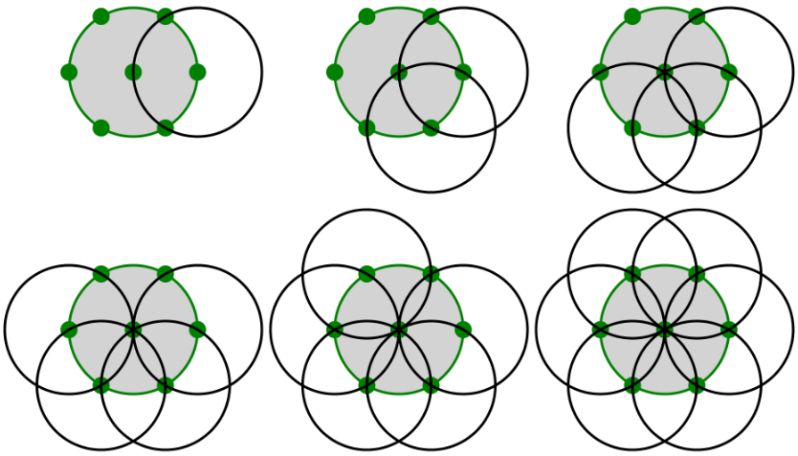


Figure 4.4: Perfect distribution of neighbors.

4.2.4 Mutually exclusive neighbors

Another promising approach is for a mote to choose its initial neighbors by selecting motes that are mutually exclusive from each other. In other words, if center mote C has neighbors A and B, they are chosen to be in the same LNET as C only if A is not a neighbor of B and B is not a neighbor of A. By choosing neighbors that are mutually exclusive, we can be assured of a well distributed set of neighbors, with paths leading towards a wide variety of targets. This is a fairly efficient method with only a small amount of overhead required, especially if the needed neighbor information can be piggybacked into some other required transmission. As illustrated in figure 4.4 six neighbors will fit perfectly distributed around the perimeter at distance r with centers that are distance r apart, however the perimeter motes each have two neighbors, one on either side, so the maximum number of mutually exclusive neighbors a mote can have is 5.

However, once a single set of mutually exclusive neighbors is chosen (5 or fewer), there is no clear approach for how to proceed. And in our tests using various means to choose neighbors beyond this first set, the turn taking method described below consistently produced better results.

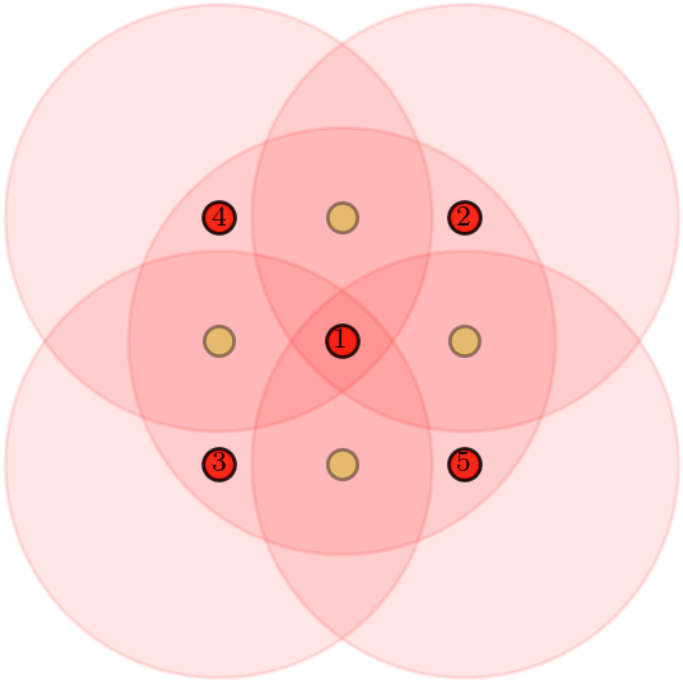


Figure 4.5: Mote 1 and mutually exclusive neighbors 2,3,4,5.

4.2.5 Turn taker partitioning algorithm

Of the methods we tested, the turn-taking algorithm results in the most accurate partitions, producing LNETs whose mean densities minimize deviation from our objective function (equation 3.4). In the turn-taking algorithm 4.7, a mote m must first choose the partition it will join j where $0 \leq j < \varphi$, the choice is based on m's own survey results, or by random choice. Next the mote ranks its neighbors by signal strength, then it proceeds to assign its neighbors to partitions that make the most sense from its own point of view. Mote m begins with its least desirable neighbor mote m_0. Mote m informs its m_0 of its new partition $(j + 1 \mod \varphi)$. If m_0 has already been assigned to a partition the mote moves to $m_1, m_2, \cdots m_n$. Once a neighbor m_x has accepted its assignment it becomes the neighbor's turn to execute the same process. When m_x returns, m proceeds to assign m_{x+1} to partition $j + 2$ and again defers to m_{x+1}. This process continues until all of m's neighbors have been assigned partitions.

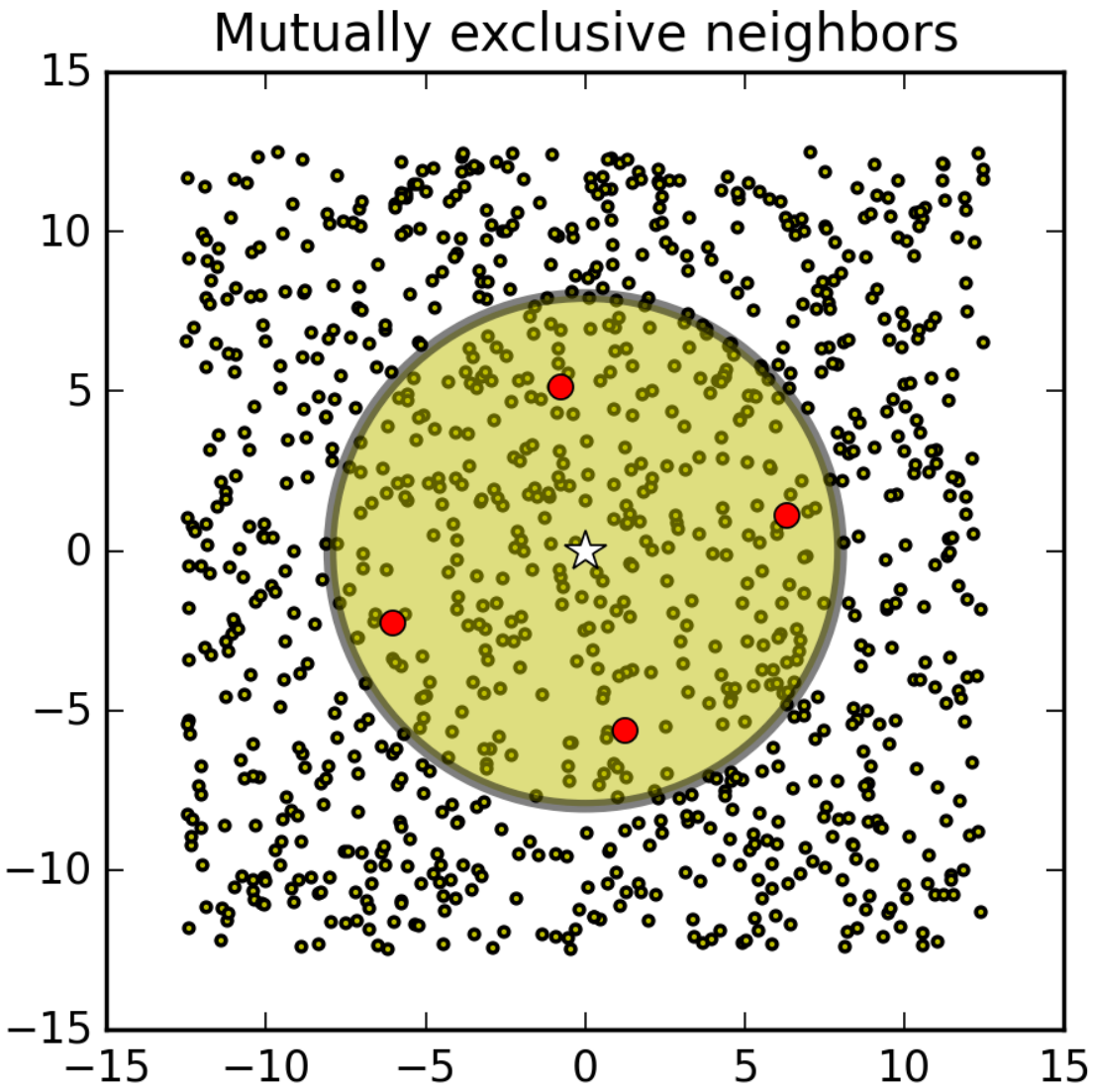


Figure 4.6: Mutually exclusive neighbors chosen by center node.

procedure TURNTAKER($mote, j, \varphi$)
    $mote.partition \leftarrow j \mod \varphi$
    SORT($mote.neighbors, RSSI$)
    for $neigh \in mote.neighbors$ **do**
        if $!neigh.partition$ **then**
            TURNTAKER($neigh, j + 1, \varphi$)
        end if
    end for
end procedure

Figure 4.7: Turn-taking partitioning algorithm

The performance of linear turn taking is markedly better than random partitioning as illustrated in figures 4.8 and 4.9. The data used in these images comes from generating motescapes, each with dimension 1000x1000 and containing 1867 randomly distributed motes with $r_0 = 75$. We perform 100 experiments for each partitioning method, during each experiment a motescape is created and its motes are randomly positioned, and each mote discovers its neighbors. At this point the motescape is partitioned into three by the partitioner, either random or turn taker and the mean density of each of the resulting partitions are measured.

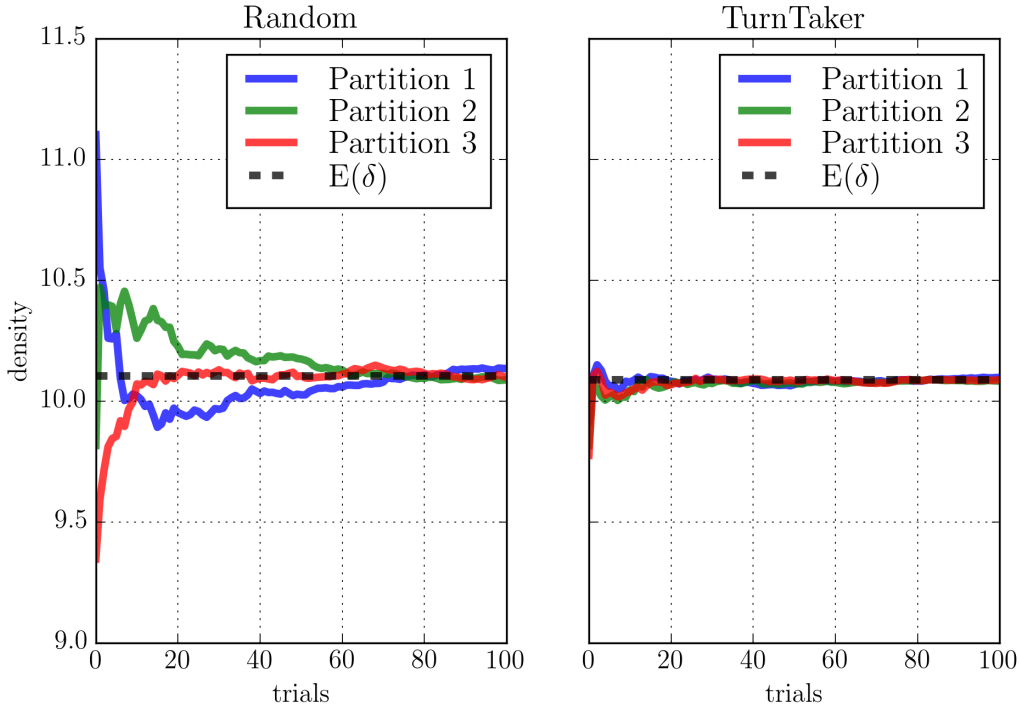


Figure 4.8: Density measurements when creating three partitions.

The data for figure 4.8 comes from 100 trials in which a set of motes is partitioned into three LNETs and the density of each is measured. Clearly the turn taker algorithm produces partitions whose densities are more consistent and have less variance as compared to partitions created with the random partitioner. Figure 4.9 shows the distribution of densities within partitions created by turn taker and Random partitioning. The degree of motes in turn taker's partitions have a narrower distribution

and thus a lower standard deviation with fewer outliers than those create by the random partitioner.

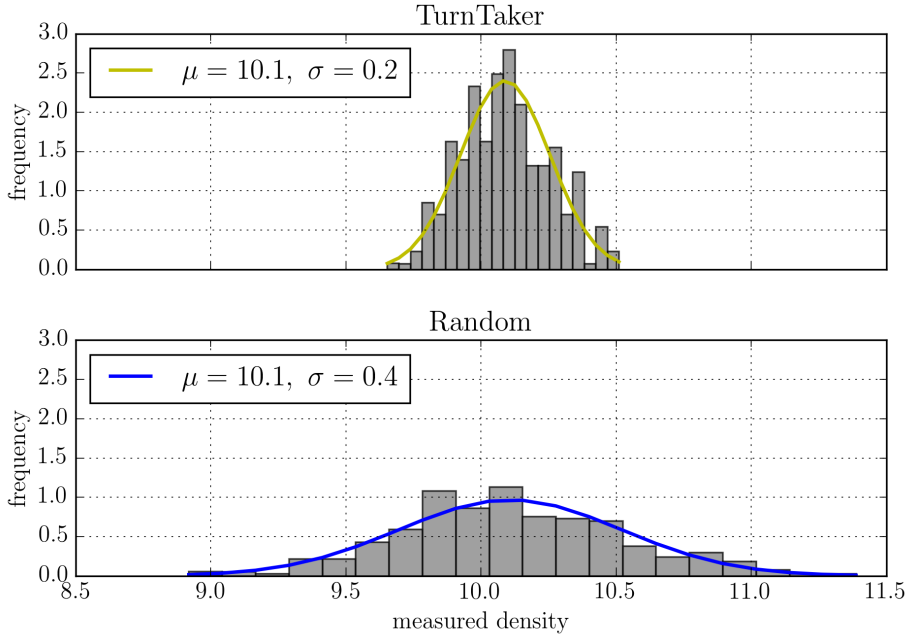


Figure 4.9: Density measured in partitions created using turn taker and Random partitioning methods.

The turn taker algorithm as described in figure 4.7 expects a single starting point and proceeds in a linear recursive fashion which is incongruous with the notion of an asynchronous distributed environment. In section 5.3.5 we give a modified version of this algorithm (figure 5.2) that works in a distributed environment by allowing motes to initiate partitioning asynchronously and requiring motes to wait for a brief random period between attempted partition assignments.

4.3 Proactive redundancy factor manipulation

In section 3.5 we discuss the use of configurable radio transmission power to manipulate the density of a motescape, by extension we can utilize this to influence the redundancy factor. The redundancy factor is based around the idea of a minimum

required density so if we are able to sufficiently increase mean density by extending transmission radius we can likewise increase the redundancy factor. Conversely it might be necessary to reduce transmission power in order to lower power usage in motes, in this case mean density will be reduced and possibly the redundancy factor.

4.4 Conclusion

In this chapter we discuss various algorithms for partitioning a dense motescape as well as some of our own hypotheses and findings. Utilizing the density approximation from chapter 3 to determine how many partitions to create we develop and present the turn taker algorithm for partitioning. This is the partitioning method we found to produce the best results by creating partitions such that each covers approximately the same area, has similar total population, and mean degree that approximates the ideal.

Chapter 5

Protocol

5.1 Overview

A defining characteristic of distributed sensor networks such as Cyber Physical Systems (CPS) is autonomous operation [10, 7, 34, 56, 57]. In order to be useful, motes that make up a sensing network must first discover and organize themselves after which they can run higher level protocols to discover network resources and determine routes for communicating with those resources.

5.1.1 Motivation

In chapter 1 we describe how the protocol described in this section addresses needs associated with scalability and cyber security (1.1. Identified needs and requirements) by responding to fluctuation in bandwidth demand as a result of changes in population density (1.5.2. Scalability) and improving availability of systems and data (1.5.3. Cyber security). The previous chapters 3 and 4 established the methods for deter-

mining density and partitioning motes that are necessary in order to implement this protocol.

5.1.2 Definitions

Channel

A communication channel is described in appendix A.1 as a "range of radio frequencies used to convey information between motes using a radio transmitter and a receiver." The number of channels and specific frequencies available depend on the type of radio. Today's prevailing standards for computer radio communication are IEEE 802.11 (Wi-Fi) and IEEE 802.15.4 (ZigBee) designed for low power and embedded devices. The number of channels and their frequency ranges used by these standards depends on the country in which the devices are sold and the version of the standard. Various iterations of 802.11 involve radio frequencies in the 2.4, 3.6, 5, and 60 GHz frequency bands, the 2.4 GHz range is divided into 13 or 14 channels depending on location[58]. The low power 802.14 standard provides 16 channels in the 2.4 GHz range, and allows use of some lower frequencies in other countries[59].

To execute this protocol all motes must first tune to a common **base channel** which we denote as c_{base}. If c_{base} is not explicitly defined motes should choose the first numerical channel available, by convention channels are numbered sequentially beginning with 1 and representing frequencies ordered from lowest to highest. Whenever a mote finds channel c_{base} to be unusable or deserted it should configure C_{base} to the next available channel and try again.

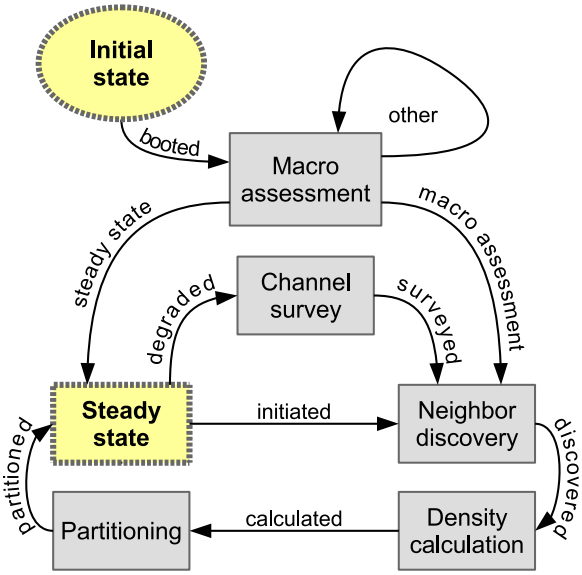


Figure 5.1: States and transitions.

Protocol description

A mote that has joined the CPS will proceed through a set of states that fall within two operational categories termed the *setup phase* and the *steady state phase*. During the setup phase communications channels are evaluated and ranked if needed, neighboring motes are located, network density and sensor exposure are calculated and finally motes choose a channel to communicate on and a logical network (LNET) partition to join. When the setup phase ends motes transition to steady state mote and fulfill the function for which they were created. While in the steady state phase system change events may occur which cause the motes to re-enter the setup phase, these are described further in section 5.2.3.

The states and transitions that make up the LNET protocol are represented in figure 5.1 and briefly described below, an in-depth description of the protocol states is given in section 5.3.

Operational Phases

1. Initial state

2. Setup Phase

   (a) Determine the current "macro" state (sec 5.3.1)

      i. If steady state, choose a channel and transition to steady state

      ii. If macro assessment, wait for time T and begin discovery

      iii. If motes are in any other states, wait until motes are in one of the two previous states

   (b) Discover all neighbors within a single hop

   (c) Determine overall population density

   (d) Select channel to become member of

3. Steady State Phase

   (a) Operate for intended purpose

   (b) Initiate channel survey in response to degraded frequencies

   (c) Initiate LNET formation in response to topology changes

5.1.3 Contribution

We give an algorithm and set of states by which a set of motes can partition themselves into functionally independent logical networks providing scalability and improved resiliency.

5.2 The LNET protocol

The LNET phase encompasses the bulk of our contribution requiring motes to assess and discover their environment, discern population density and form partitions. In

this section we discuss details and procedures necessary for distributed operation and dissemination of data required for motes to be able to form efficient partitions based on accurate information.

We first discuss how the state diagram in figure 5.1 applies to individual motes as well as the group of motes as a whole. We then discuss the means of coordinating a large population of motes to enable distributed calculation and communication. We then go into details concerning the operation of our protocol for each of the protocol states.

5.2.1 Micro state and macro state

The states shown in figure 5.1 apply to each individual mote in a given motescape. However since the motes move through the LNET phase as a group, the same state diagram can also be used to describe the state of the LNET process as it applies to all motes in M. We refer to the view of these as the **micro state** in an individual context and the **macro state** in the group context.

As an example if all motes are in the NEIGHBOR DISCOVERY state, then the state of the LNET is also that of neighbor discovery because the micro states and the macro state are synchronous. But once some mote m transitions to DENSITY CALCULATION, the micro states are no longer homogeneous and so the state of the LNET is undefined with respect to the state diagram in 5.1 until all individual motes are once again in the same state.

When determining the macro state, we exclude motes that are in the INITIAL STATE MACRO ASSESSMENT STATE. Because individual motes may be added at any time, these motes are not yet part of the CPS. Thus if all the motes that have joined the

CPS are in the same state, the existence of motes that are not yet active will not cause an undefined macro state.

5.2.2 Coordination of motes

As with any cooperative distributed protocol there is a certain amount of coordination required between the distributed participants. In appendix A.3 we give a list of waiting period definitions that we use or foresee being useful. These waiting periods should ensure sufficient time for completion of distributed processes such as message propagation and channel switching. The actual amount of time represented by these variables is highly dependent on a wide variety of factors such as deployment environment, software implementation and mote hardware.

We mitigate the need for maximum transition and calculation periods by allowing motes to *catch up* with state transitions initiated elsewhere. Generally a mote m will operate in a given state for an alloted time period after which it will transition into the next state and transmit a signal. The catch up ability lets m truncate its remaining time in a given state if it sees a message indicating other motes have already begun to transition to the next state. In this way, some of the error in synchronization caused by message propagation or clock drift can be mitigated. Moreover the time in which the macro state is undefined is minimized as is dependency on clock synchronization between motes.

5.2.3 Triggering the LNET phase

The setup phase is initially triggered by mote deployment, and may be subsequently initiated either periodically or due to external factors as enumerated below:

LNET formation triggers

1. Period expiration (W_s)
2. Density change
3. Topology change

Period expiration

In appendix A.3 we describe various waiting periods to ensure sufficient time for processes such as coordination among nodes, message propagation and channel switching. The actual amount of time represented by these variables is highly dependent on a wide variety of environment and implementation specific factors.

The **initial deployment waiting period** W_d begins from the time deployment begins and should encompass enough time to allow deployment to complete. This period should ensure that all motes begin neighbor discovery at about the same time and prevents motes from wasting energy on multiple discoveries due to incomplete deployment.

Upon initial deployment motes should wait time W_d before beginning the discovery phase. The waiting period should ensure adequate time for mote deployment. Once this period has elapsed for each mote it should enter into the discovery phase. Instead of a *period* of time, W_d may represent an *absolute* time after which discovery phase can begin.

The **setup initiation period** W_i is a waiting period of time that begins whenever the decision to re-form the network has been made due to changes in topology or density. This allows for all motes in the network(s) time to be notified that the network is re-forming and revert to the base channel. This time period should be long enough to encapsulate the **timeout period** W_o so any motes that were unable to receive the notification message can time out and switch back to the base channel, however in networks where motes have long periods of inactivity this may not be practical.

It may be advisable to establish a **transition period** to give motes a buffer for transitioning between phases. This period should encompass the amount of time required for motes to perform state changes as well as accounting for clock disparity among the various motes. This ensures that a mote is not caught between two different phases.

Density change

Dynamic population change may result in inaccurate density and/or sub-optimal LNET composition. These changes can occur as a result of factors such as node movement, addition of nodes or removal of nodes that have ceased to function or exhausted their power supply. If the number of motes in the network changes sufficiently to warrant an increase or decrease in the number or makeup of partitions then network motes may initiate network re-formation via control messages.

Wireless topology change

Changes in network topology may cause an existing LNET to experience communications error beyond a given threshold. A possible cause is external interference that was not previously present. In this event some mote in the affected network must initiate the re-formation process.

5.2.4 The LNET initiation process

When a mote decides to initiate the process of forming the network, it should record the time as T_f and then broadcast a command to its neighbors informing them of the decision. The command for re-forming the network should include the T_f so that the amount of time motes wait before initiating the discovery process W_f can be

calculated relatively by all motes in the network. This enables propagation of the notification message and allows all motes to begin the formation process together reducing time spent in non-operational states.

Once the motes in network n decide to initiate a reconfiguration, each mote should complete the following steps:

1. Notify neighbors by broadcasting the command to re-form the network
2. Notify another network by randomly choosing another network, switch to its channel and broadcasting the command to re-form the network
3. Switch to the base channel and begin discovery after a waiting period

Notify neighbors: Before new LNETs can be forged, existing LNETs must be dismembered so all motes can participate in the process. Upon receiving a command to re-form the network a mote must rebroadcast the command to its neighbors preserving the original time stamp T_f from the command it received.

Notify another network: After notifying its neighbors, mote m must randomly choose a channel c_r, switch to channel c_r and broadcast the command message to re-form the network.

Switch to base channel: The process of neighbor discovery requires that all motes are communicating within the same channel. All motes are configured with a common base channel c_{base} where neighbor discovery is attempted, or if c_{base} is not explicitly defined motes should choose the first channel available $c_{base} = c_0$.

When discovery is initiated, all motes not already operating on channel c_{base} must switch their radio to use this channel, allow time for transition and then begin the discovery process.

If discovery cannot complete on channel c_{base} within time W_x, motes should update the base channel to the next available channel $c_{base} = c_{base} + 1 \bmod |C|$. The mote must then broadcast a message to alert neighbors of the new base channel and switch the radio to the new c_{base} and try again.

Whenever a new mote is added to an existing network, it should attempt to discover neighbors in c_{base}, if no response is received within the time W_x the mote shall iterate through channels in the manner detailed above.

5.3 States

5.3.1 Macro assessment

When in this state a mote will assess the macro state of the LNET to determine whether it is joining an operational system, or waiting with all other motes to begin LNET formation. A mote will continue in this state until the macro state is found to be either STEADY STATE or MACRO ASSESSMENT.

If the macro state is determined to be STEADY STATE, the mote will join the existing LNET. This can happen upon initial deployment if motes are added dynamically rather than as part of a mass deployment, or if LNET formation occurs before all motes are deployed. Similarly if a mote has to reboot due to error it will likely need to join an existing LNET. Additionally if a mote has been absent for a period of time for reasons such as power conservation must re-assess the state of the system.

Macro state → subsequent action

- STEADY STATE → join the existing LNET
- MACRO ASSESSMENT → begin the LNET formation protocol
- *Others* → monitor status until one of the above actions can be taken

Status query beacon and response

When a mote first enters this state it broadcasts a status query beacon. This beacon indicates the mote is attempting to determine the macro state and any mote receiving the broadcast must send a response if and only if it is operating in STEADY STATE.

A query response indicates that the LNET protocol has taken place, and that the new mote should join an existing partition. The query response will include the information gathered from the LNET protocol which the new mote will need to join an existing partition including the redundancy factor and lists of active and vetoed channels as described in section 5.3.2.

Macro state determination

Motes that are assessing will note query beacons as well as query responses. If the only transmissions seen for a specified period are status query beacons, the macro state is assumed to be MACRO ASSESSMENT and the mote may transition to NEIGHBOR DISCOVERY.

If any motes respond with a status query response packet, the macro state is assumed to be STEADY STATE and the mote should choose a partition to join and transition to steady state. Since the populations are expected to be large, occasional additions should not upset the overall balance and random choice is acceptable, however a survey could be undertaken in order to find the best fit.

5.3.2 Channel survey

A mote will enter this state only if the channel it is using for communication becomes degraded as decided by itself or its neighbors. Reasons for this may include outside devices that have begun using the channel either cooperatively where the outside devices are aware of the existing motes or non-cooperatively in which the outside devices are not aware of their intrusion. Cooperative use results in diminished bandwidth, where as non-cooperative use causes interference. During this state all channels in C will be evaluated and then ranked according to desirability, and the set of channels eligible for use during the partitioning phase (section 5.3.5) will be derived from these rankings.

When the survey begins each node will randomly choose a channel from C to evaluate. It will then set its radio to listen on the selected channel and count the number of interrupts that indicate signal reception. Once the channel evaluation time period W_e has expired the node should return to base channel. The count of received decodable (congestion) and un-decodable (interference) packets will be included in the *beacon* during the upcoming discovery phase so that we can build a network wide consensus of most suitable channels without requiring an additional global data exchange.

Once the tables have been created, motes will simply keep a sliding window average of the number of un-decodable packets received per minute. If interference worsens such that quality is far enough below an unused channel then the setup phase will be initiated and networks reformed.

Managing topology changes

If a channel that was vetted by the channel survey becomes unsuitable after being chosen by a set of motes, the mote making the determination of unsuitability must

choose a new channel and veto the current channel. Given a list of channels that are already allocated to an LNET C_A, and a list of those that have been vetoed C_V, a mote may veto its own channel c_m using the algorithm given below.

Vetoing a Channel

1. Remove the channel from the list of allocated channels $C_A = C_A - c_m$ and add it to the veto list: $C_V = C_V + c_m$.

2. Choose a new channel by modularly incrementing the existing channel until an available channel is found: $c_m = \{(c_m + 1) \mod |C| : c_m \notin (C_V \cup C_A)\}$.

3. Broadcast a *veto* message to neighbors indicating the new channel C_m.

Any mote initially receiving the veto notification should update its own C_A and C_V tables, re-broadcast the message to its neighbors and switch to the new channel. If the new channel is found to be occupied, updated tables and restart the process.

5.3.3 Neighbor discovery

Neighbor discovery is accomplished by broadcasting messages over the common base channel c_{base}. Discovery initially takes place en-masse either as the first phase of the LNET protocol or immediately following the channel survey. Discovery may be directly entered by new motes being added to the network that were not present during the initial deployment or by motes that have become isolated from their former neighbors.

Upon entering the discovery phase, each mote m will send an advertising beacon and await responses.

Discovery beacon

The use of wireless communication within a mote is expensive. Not only is it relatively slow but it typically represents the largest energy demands. It is therefore imperative that whenever a broadcast is to be undertaken that any opportunity to consolidate additional data be taken advantage of in lieu of requiring a separate transmission. In this spirit the discovery beacon may contain additional data from previous operations or for use in future operations.

If a mote previously completed a channel assessment, it will include its rankings and any vetoes. Additionally if the algorithm used during the PARTITIONING state is a simple random choice, then including a sufficiently large random value in the discovery beacon will eliminate the need for a second neighbor discovery phase upon transitioning to STEADY STATE.

Discovery beacon contents

1. Any channel rankings compiled during the assessment phase (section 5.3.2).

2. A unique ID by which a mote can be distinguished.

After a mote sends its discovery beacon, it will continue to listen for other beacons and acknowledgments for time W_n or until any of its neighbors transitions to the next state.

Beacon acknowledgment

Motes that receive a discovery beacon will respond with a neighbor acknowledgment message. The existence of this acknowledgment means the responding mote x has added mote m to its neighbor table and that m should likewise add x to its own neighbor table.

5.3.4 Density calculation

We use information about mote density to calculate a ratio of *measured density* to *ideal density* in order to determine the number of channels needed to divide our collision domain as well as the number of redundant sensor networks that the motescape can support. These density calculations can also be used to find trouble spots where motes are distributed too closely or too sparsely. The details of density calculation are described in section 3, the formula we use is given in equation (3.12) which is a refinement that mitigates the problem of edge effect giving a more accurate estimate of density.

5.3.5 Partitioning

Each node must now select one of the available channels to communicate on. The method a mote uses to determine its channel is its partitioning method. In section 4.2 we discuss a number of approaches for partitioning in which motes adopt or are assigned to a partition, or LNET. Since each LNET is assigned a unique communication channel, by executing the partitioning algorithm during this protocol phase motes will be associated with a channel.

Random partitioning

During neighbor discovery each mote included a random or unique identifier in its beacon message. Using a simple hash function we can deterministically map these identifiers into the set of channels selected for partition use. This has the advantage that each mote can now discover its true neighbors by applying the same hash function to the its neighbor's identifiers which negates the need for a second discovery phase.

```
procedure TurnTakerDistributed(lnet)
    if self.partitioned then
        RETURN
    end if
    self.partitioned = TRUE                          ▷ Prevent re-execution.
    if !lnet then
        self.partition ← random(0, φ)                ▷ Choose random LNET.
    else
        self.partition ← lnet                        ▷ Use neighbor assigned LNET.
    end if
    SORT(self.neighbors, RSSI)                       ▷ Order by proximity.
    j ← self.partition
    for neigh ∈ self.neighbors do
        j = j + 1  mod φ
        TX(neigh, TurnTakerDistributed(lnet = j))
        WAIT(W_a)                    ▷ Pause between assignments (take turns).
    end for
    self.channel ← C{self.partition}                 ▷ Switch to new channel.
end procedure
```

Figure 5.2: Turn-taking distributed implementation

Turn taker partitioning

We presented the basic turn taker partitioning algorithm in chapter 4, as described in figure 4.7 the algorithm is easy to follow and illustrative of how this method works. However this version is not suitable for distributed operation, so in figure 5.2 we give a variation of the basic algorithm that works within a distributed environment, this is the algorithm we use for our full simulations.

Motes should record all neighbor assignments it sees during turn taking in order to maintain its neighbor tables. By doing this motes can avoid having to undergo a second neighbor discover phase once channels have been selected/assigned.

5.4 Conclusion

In this section we utilize the density approximation from chapter 3 and the partitioning protocol developed in chapter 4 in order to give the states and algorithms necessary to allow a set of motes to maintain a mean density value within a given tolerance. The viability of this protocol is borne out by our own implementation and subsequent simulations using the Python programming language and the MatPlotLib suite of scientific and statistical analysis software. We present the results of our simulations in chapter 6.

Chapter 6

Simulation

6.1 Simulation

We use custom simulation software written in Python and using the NumPy and MatPlotLib modules from the popular SciPy stack for statistical analysis and graphing. Our simulations are controlled by a discrete clock process, and include elements that are stochastic and dynamic. We use simulation to test our hypotheses, compare different approaches, calibrate algorithms and validate assertions.

Questions addressed with simulation

1. Does density have a significant effect on contention for bandwidth in a wireless network?

2. Can this be effectively mitigated using multiple channels?

3. Is there an effective and efficient method for partitioning an ad-hoc wireless network?

4. What measurable improvements are gained within the context of a complete protocol?

5. Is there an effective and efficient method for calculating the density of an ad-hoc wireless network?

6.1.1 Purpose

The decision to code our own network simulator stems from requirements that are specific to our study of creating and operating very dense networks of devices that communicate using radio, a limited range broadcast medium. In particular we wanted to gather statistics on the behavior of motes as they face heavy contention for access to shared wireless communications frequencies. We investigated several popular network simulation solutions but found them unsuitable for our purpose.

6.1.2 Existing simulation packages

We hoped initially to use existing simulation software to test our hypotheses and so we investigated the suitability of several popular packages but encountered fundamental problems with each with regards to the particularities or perceived needs of our research.

NS-3

We initially looked at NS-3, the well known modern successor to the venerable NS-2 simulator[60]. We quickly rejected NS-3 because it had no provision for simulating low power wireless communication which we had initially perceived as a requirement.

Castalia

Castalia is a Wireless Sensor Network simulator maintained by an Australian research institution with the acronym NICTA, the meaning of which isn't entirely clear[61]. Castalia is built on top of OmNet++ which is a popular generic simulation platform[61]. We needed to measure thousands of fairly anonymous and random devices however Castalia seems to expect specifically defined and configured nodes with individually specified connections.

TOSSIM

Hoping to leverage familiarity with TinyOS Alliance's "TinyOS" operating system we attempted to create a basic simulation using the TinyOS Simulator TOSSIM. After some code investment we found that TOSSIM does not simulate radio propagation, but provides an abstraction for peer to peer connections[62, 63].

Octave

Finally we decided upon a more generic approach of using GNU Octave[64]. Octave is an open source programming language that has extensive graphing capabilities and is largely compatible with MathWorks' MATLAB. This approach worked to a point, however it soon became clear that a complex simulation would require a fully featured programming language.

Python and SciPy

Finally we discovered the scientific computing packages available for the Python programming language, and in particular the SciPy stack[65]. SciPy encompasses several

core packages, the most relevant for our needs are NumPy and MatPlotLib. Like Octave, NumPy provides features modeled on MATLAB which facilitated porting our Octave code into Python while MatPlotLib contains a powerful graphing library for visualizing simulation results.

6.2 Simulator models

Whenever possible we have modeled our simulated configurations on existing protocols and standards such as the low power radio standards defined in IEEE 802.15.4[59]. Whenever unable to find or procure the required configuration specification, we adopt relevant industry or academic models.

6.2.1 Code structure

Our simulator code is broken into three main packages named PySim, Graphler and CPSSim this structure is represented in figure 6.1.

PySim contains abstractions and class definitions for Fields and Nodes as well as common utility functions such as saving and loading data and density calculations.

The Graphler module is focused on executing simple experiments, compiling and displaying the results. This module has simplified Field and Mote classes and contains numerous experiments in which motes have limited or no interaction. This module also contains a front end to the MatPlotLib graphing functionality aimed at producing visually consistent plots.

The CPSSim module executes only a few types of simulations as compared to Graphler, but its simulations are more complex requiring session based mote interaction, protocol implementations and multi-hop routing. The Graph and Mote classes

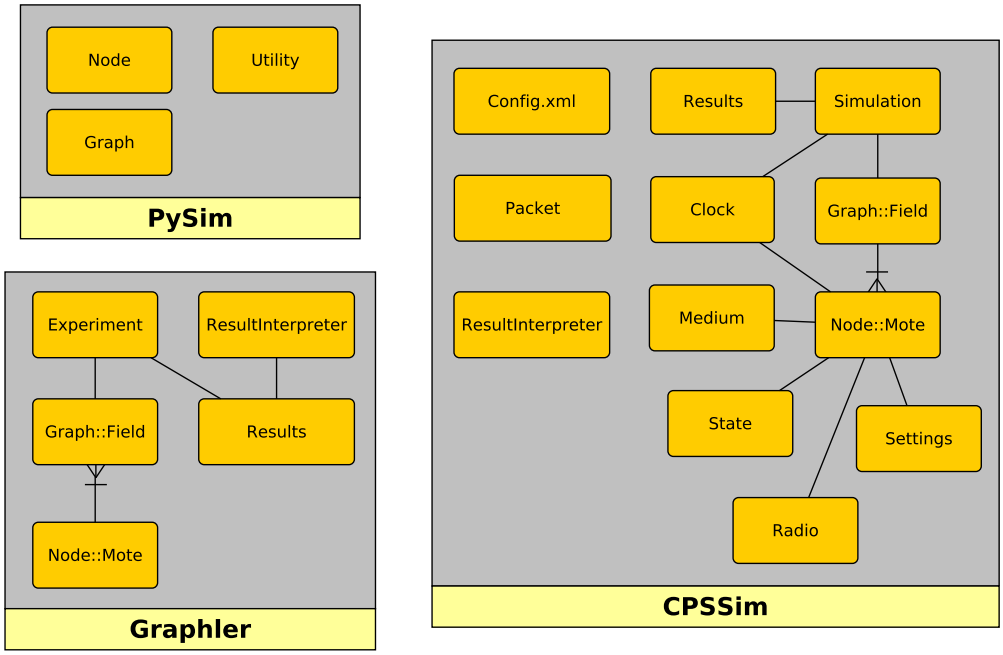


Figure 6.1: Structure of simulator code

are granular and composed of a number of independent objects such as radio, packet, medium each of which has its own configuration and state.

6.2.2 Event model

We initially coded our simulator as a trace-driven simulation model in which every mote would execute as an independent thread. As we attempted to simulate large mote populations this model immediately proved to be untenable. To achieve the needed mote densities we re-factored the code so that all simulated activity is controlled by a global clock, transforming our simulator to a discrete-event simulation model.

Our simulations progress through discrete events controlled by a clock that has a time granularity of one millisecond, or 1000 ticks per second. This time definition lets us use realistic and recognizable values for simulated events such as message

propagation, boot times and timeout errors while allowing simulation of sufficiently large mote populations. The most intensive simulation we ran consisted of around 2500 motes having average degree between 40 and 50 running for a simulated 30 minutes which equates to 1.8 million *clock cycles*. Each time the clock ticks any events that are due are executed. These are mostly functions within individual mote objects, but can also be callbacks to the simulator for event notifications.

6.2.3 Distribution model

We assume a network of static motes distributed in a uniform random manner. A static network is a network in which the nodes are not mobile. Mobile nodes result in an ever shifting topology which requires frequent updates to keep routes and neighbor tables current. Mobility in an ad-hoc network can improve overall capacity, but at a cost of complexity [29].

6.2.4 Failure model

Communication can fail in two ways, an inability to get a successful CCA and interference. In order for a mote to begin broadcasting it waits until it detects no other motes are transmitting within range, this is a successful CCA result. The CCA for a mote will fail if *any* of its neighbors are in the process of transmitting. In the event of CCA failure the mote will back off then retry until either the channel is clear or the number of consecutive failures exceeds the number of configured retries. The back off time *ccaBackoff* shown in figure 6.2 is calculated each time it is accessed using the formula specified in IEEE 802.15.4 [59] for low power radio.

A mote's reception can be interfered with if it has neighbors on opposite sides that transmit simultaneously. This is the *protocol model* introduced by Gupta and Kumar

that allows communication as long as the two motes communicating are closer to each other than any other mote concurrently transmitting on the same channel[44].

6.2.5 Contention model

In order to simulate resource contention each mote in our field attempts to transmit a small amount of random data "chatter" at semi-regular intervals to its one-hop neighbors. The between chatter messages randomly varies for each mote as shown in 6.2. This *chatter* is not multi-hop and so is only visible to neighboring nodes.

At regular intervals an *event* occurs on the east side of the motescape. The event is detected by any motes that are within sensing range. Event sensing is independent of communication channel. The sensor motes then attempt to transmit information about the event to a randomly chosen *sink* h hops away towards the west side of the field.

6.2.6 Radio configuration and modeling

To determine radio configuration we use the following documents for reference: *IEEE 802.15.4 standard for low power wireless communication*[59], the *CC2420 Technical Manual* for the popular low power CC2420 radio[66], and the *CC2420 Radio Stack* documentation for the TinyOS operating system[67] as well as the TinyOS source code itself[68].

Physical radio transmission range is defined in decibels, for simplicity we choose to use a unit of distance. As a reference we choose simulation ranges similar to examples from Bettstetter who uses a simulation field of $1000m^2$ containing 500 nodes and transmission ranges $r_0 : [70m, 100m]$. [26]

Figure 6.2: Radio configuration.

```
# How often until next chat message
chatFrequency = random(1000,1500)
# Chat duration
chatTime = random(10, 30)
# Number of retries
ccaRetries = 3
# Backoff time
ccaBackoff = 10*random(0, 2**4-1)
# Time required to send one byte
propogationDelay = 1
# Transmission range in meters
radioRange = 75
```

6.2.7 Routing model

In our simulations, a single *attempt* consists of choosing a set of p motes $m_1, \cdots, m_p$ that form a multi-hop path of length *p-1* through the motescape M. Starting with mote m_1, each mote will attempt to transmit the packet to the next mote in the path. If the packet traverses all *p-1* hops the attempt is a success, otherwise the attempt is a failure. When measuring success with redundant networks, an individual attempt can fail while the overall result succeeds. This is because with redundant networks we only require a single instance of the event to arrive at the sink regardless of how many are attempted.

6.3 Simulation control model

To validate our work we created simulation software using a stochastic deterministic model in which an area of a given size contained various numbers of simulated motes. Each mote could communicate with any of its neighbors within a range of r_0. Each mote attempts to communicate with one of its neighbors at random times once every simulated second in order to simulate an active network. We then inject a message

with a multi-hop route and measure delivery statistics such as success, failure, retries and time for end to end delivery. For comparison we then create multiple logical networks (LNETs) by assigning motes to different communication channels. For instance a motescape that has twice the ideal population of motes would be divided into two LNETs by causing each mote to communicate exclusively on one of two channels such that any random mote has an equal probability of communicating on either of two channels. Once partitioned a mote can only communicate with neighbors that are assigned to its channel thereby limiting contention for bandwidth.

6.3.1 Ideal density

The idea of ideal density (ℓ) is the mean node degree we would prefer to have in our motescape. We use ideal density to describe how dense a network is and also to determine a *redundancy factor*. For example, if we refer to densities of 1x, 2x, 3x etc... we are specifying coefficients for the ideal density indicating that the network has 1, 2 or 3 times the ideal density respectively. The redundancy factor is the number of networks we can divide our motescape into such that each network has a near ideal density.

Since ideal density varies with many factors, we use simulation with varying densities in a randomly distributed motescape to determine the density that yields the best overall performance. We used a path consisting of 5 motes or 4 hops. The results of this experiment shown in figure 6.3 indicates that in our simulated environment an ideal density value of around 0.5 yields the best combination of reliability and throughput. Lower densities may cause an inability to find a route to the sink while a higher density may result in dropped packets due to congestion.

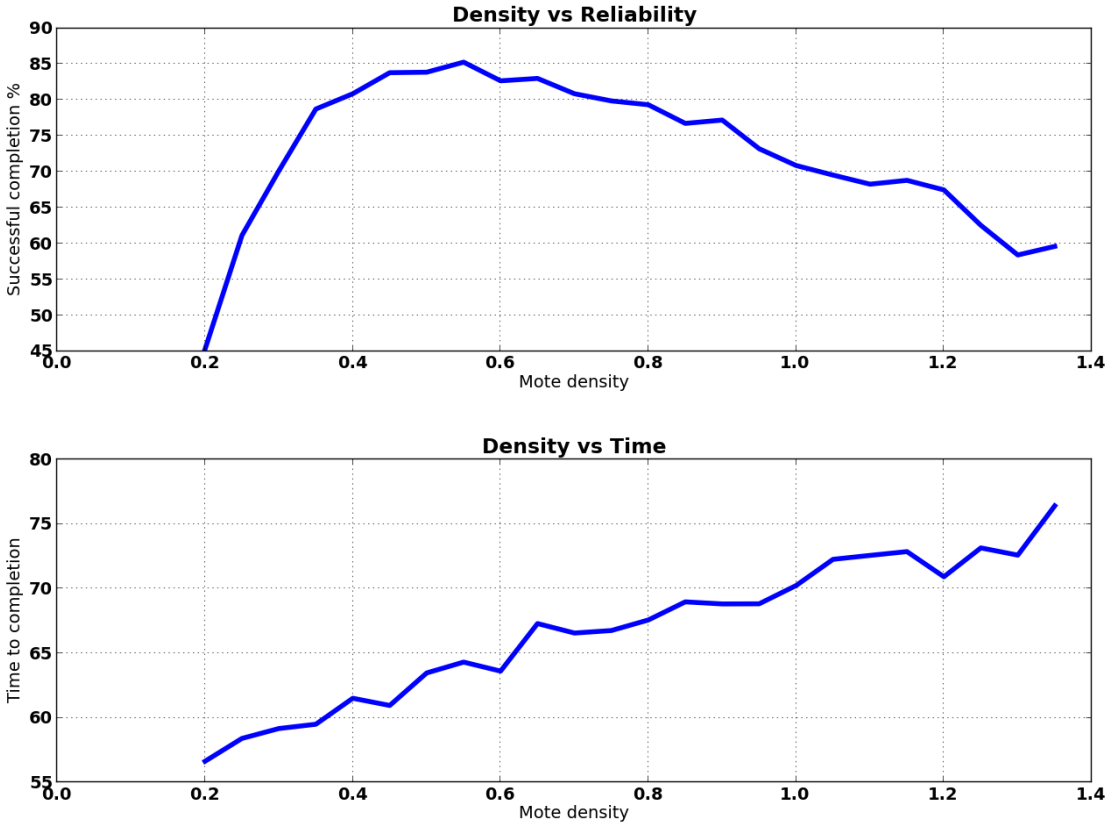


Figure 6.3: Varying density to determine an ideal.

6.4 Simulation of redundancy

In this section we show the results of a set of simulations to show the effect of partitioning on contention and reliability. The experiment used simulates an event in one quadrant of a motescape, the motes that detect the event then attempt to transmit an account of this event through some number of hops to a data sink in a different quadrant.

6.4.1 Single channel

We first simulate the basic case where a single communication channel is used for all motes, and measurements are taken for various densities. The results in figure 6.4

show that as contention for the shared channel increases we see a marked decrease in communication reliability. This is exacerbated as the number of hops required for packet delivery increase.

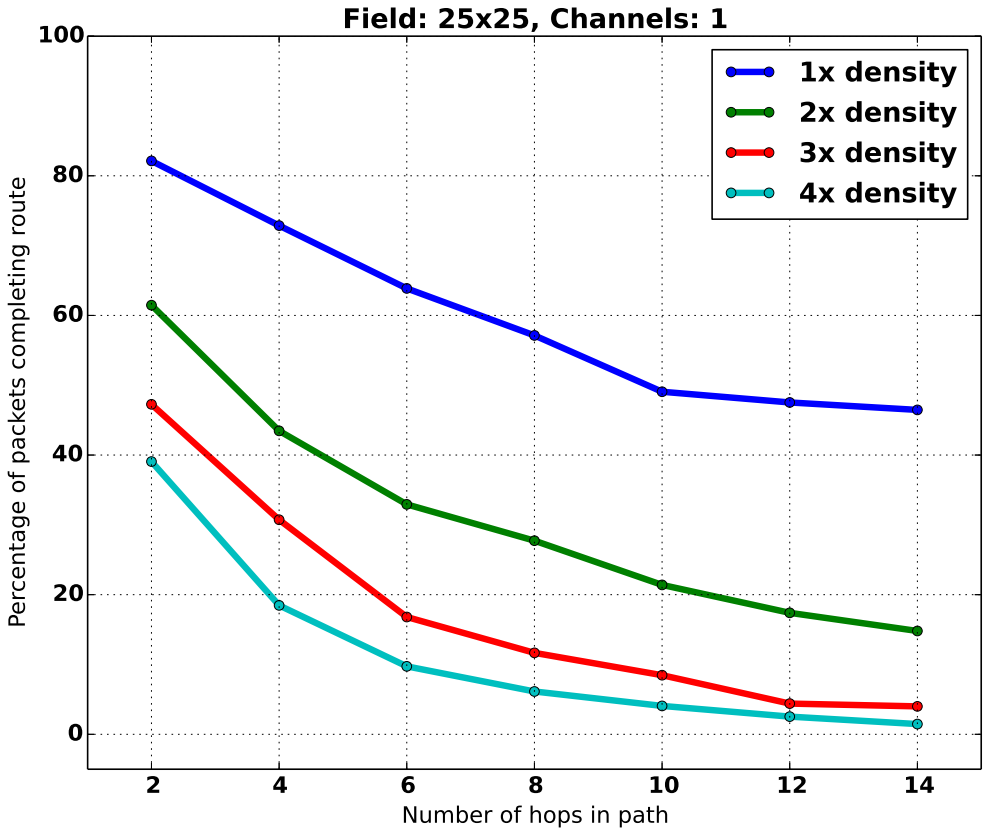


Figure 6.4: Single channel as density rises.

6.4.2 Multiple channels, independent partitions

In figure 6.5 we see that as mote density increases there is a corresponding increase in the number of communications channels available. At the beginning of a trial, each mote first arbitrarily chooses a communications channel from the pool of channels given, and only communicates on that channel.

In this simulation an "event" causes messages to be transmitted in each LNET but the results are not correlated at the data sink so all messages are treated as independent information. So if there are four LNETs and two LNETs successfully communicate an event to the sink and two do not it counts as two successes and two failures.

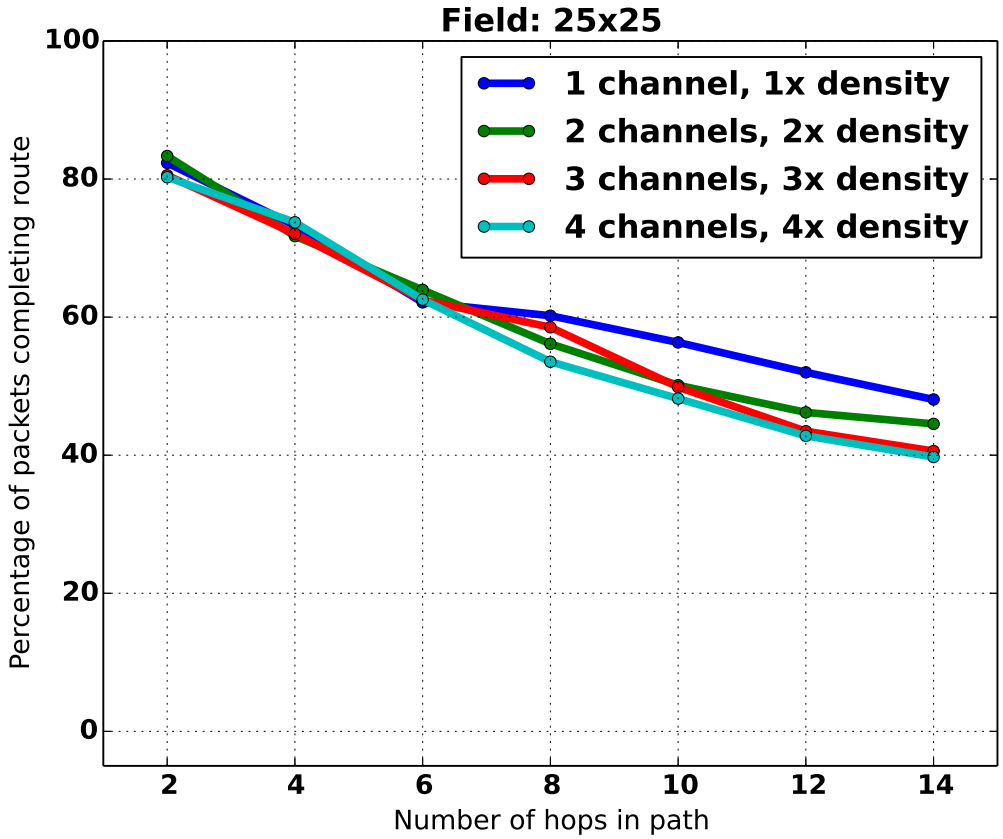


Figure 6.5: Multiple channels as density rises

The results in figure 6.5 demonstrate that this effectively mitigates the effect of contention due to rising population and each network behaves as a single channel network in a motescape with ideal density.

6.4.3 Multiple channels, redundant partitions

In our final simulation we show the results where messages are correlated at the data sink and thus each additional LNET provides redundancy. In this simulation we see that we have not only reduced congestion but exponentially improved communication reliability. The results in figure 6.6 show a motescape with 4x ideal density as we

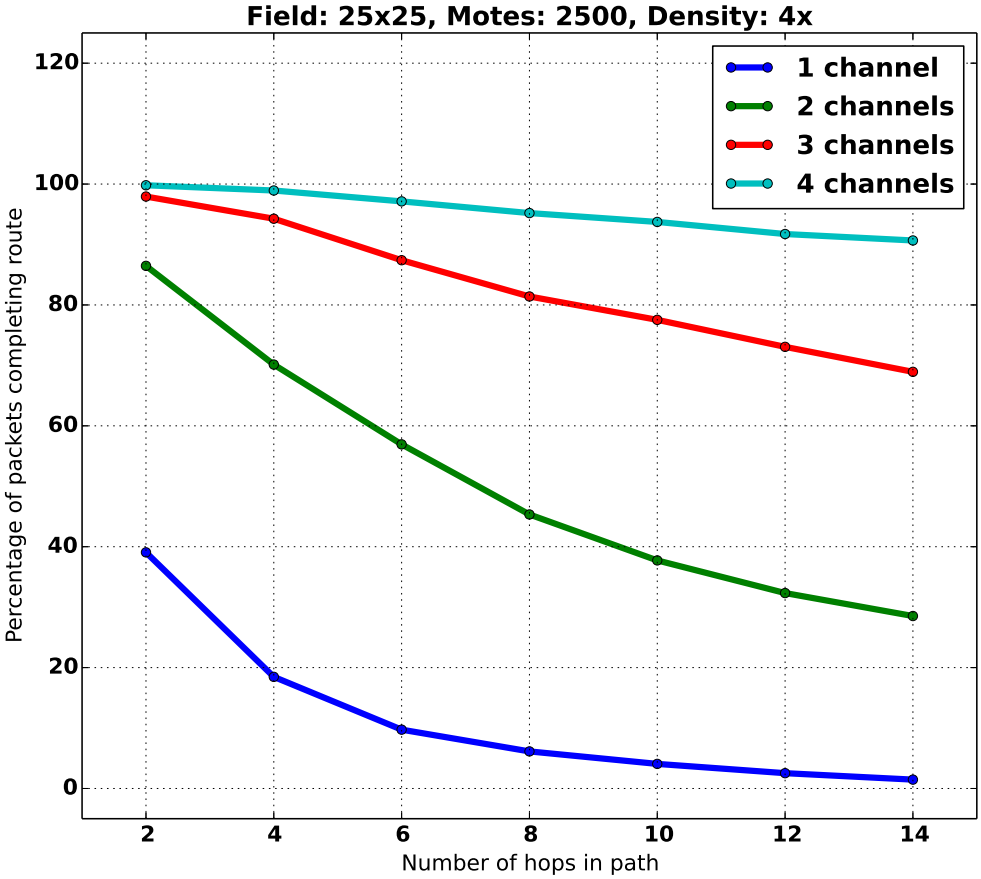


Figure 6.6: Redundant networks over multiple channels

vary the number of redundant networks from 1 to 4. Instead of treating each network as a separate entity, we count success if *any* of the redundant networks successfully transmits the results of an event over the required hops. In other words if there are 4

networks and any of them is successful we count a success, and only if *none* of them are successful do we count a failure.

6.5 Full protocol simulation

After validating the feasibility of this approach we moved on to create a full scale simulation of our protocol in which motes discover neighbors, determine density and partition themselves before switching to steady state. While in steady state motes will randomly communicate amongst themselves to generate traffic as specified in section 6.

Figures 6.7 and 6.8 clearly show the effects of increasing population density on a single partition of motes. Motes begin to experience an inability to obtain a clear channel for transmission, this causes the mote to store the data in a queue and wait to retry the transmission according to the backoff algorithm described in section 6.2.6. If a mote is unable to transmit after a number of retries it drops the packet and a failure is recorded, as the density of the motescape continues to increase the communication failures rise. Once the LNET protocol is executed the motes divide into three LNETs communicating on independent channels, at this point both the queue lengths and the transmission errors fall dramatically and remain low due to the reduced contention provided by multiple LNETs.

6.6 Conclusion

Our simulations of partitioning and redundancy clearly show the need for scaling in response to the inadequacy of single channel communication. Increasing the number of LNETs in response to rising density we see dramatic improvement in performance

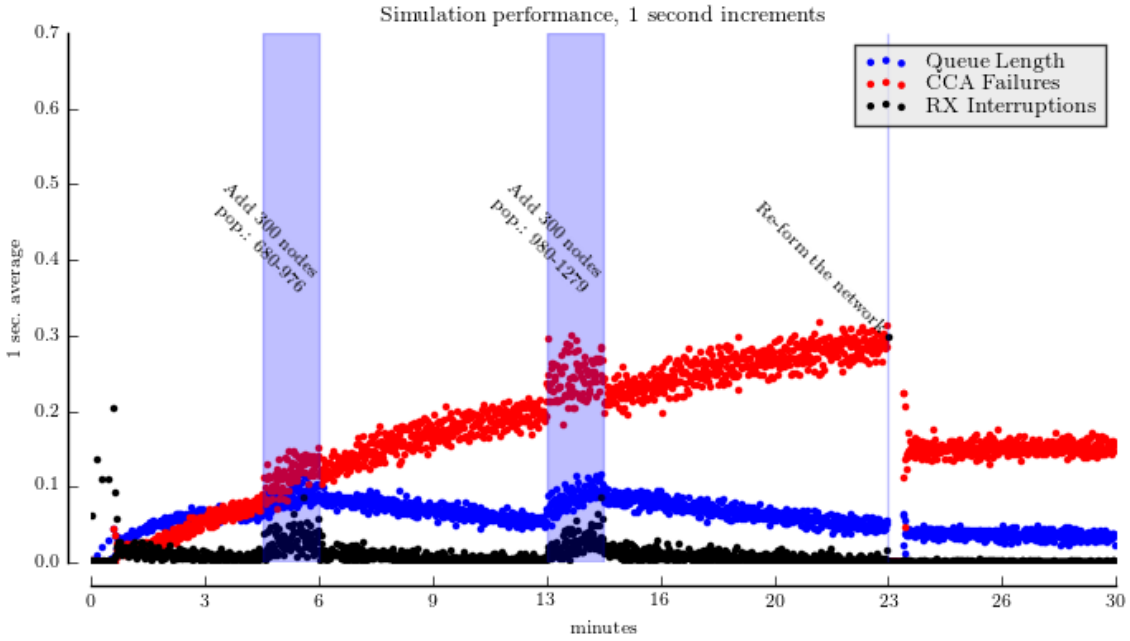


Figure 6.7: LNET scaling in response to population increase

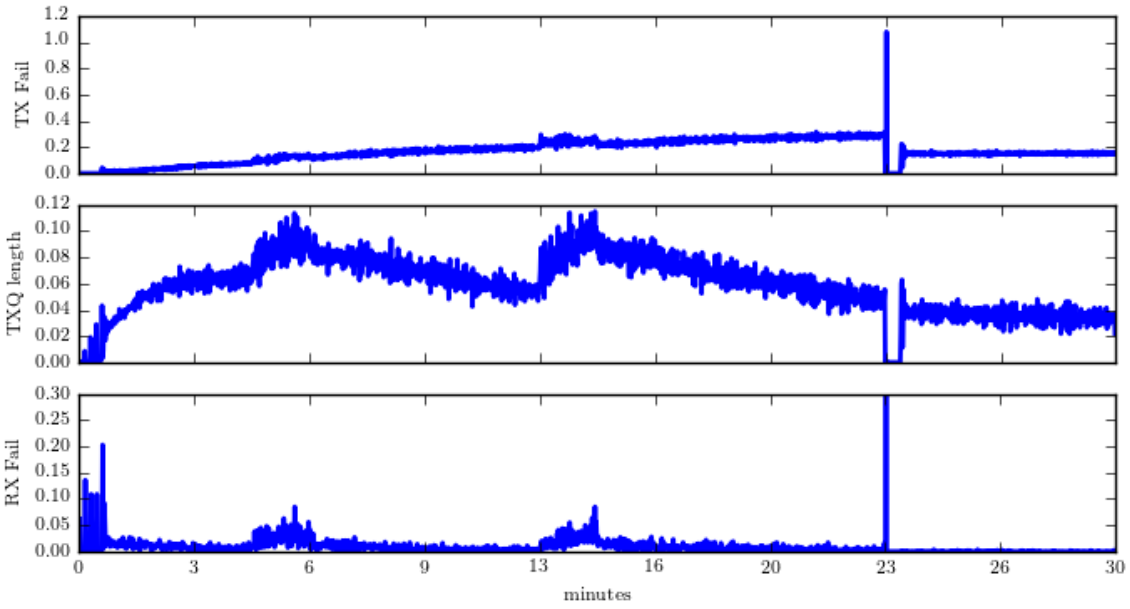


Figure 6.8: Breakout of statistics from full simulation.

approaching the case of ideal density. And finally by using the LNETs as redundant sources of data, we obtain exponential gains in communication reliability. Then our protocol simulation shows that these results can be obtained in a full simulation where motes operate independently and use distributed protocols for communication.

Chapter 7

Conclusion

In chapter 1 we identified outstanding needs in the area of cyber physical systems and laid out the contributions we make in addressing them. We proceeded in a bottom up manner first giving a means of approximating density in chapter 3, then using the density estimate as part of our partitioning algorithm in chapter 4 and finally giving a protocol in chapter 5 that allows motes to use the partitioning algorithm to divide themselves into logical networks (LNETs).

In chapter 6 we presented extensive simulation results where use of these methods confer scalability, reduced resource contention and increased reliability of communication.

7.1 Further research

Our work has shown the viability of using redundant hardware to provide a more robust remote autonomous environment. Avenues for continued research can be classified as either optimization or extension of the protocols and methods described.

7.1.1 Optimizations

Fractional channel use In heavily populated areas, a number of channels may be in varying degrees of use by devices that are not part of the motescape. In this case we would like to be able to use these channels while taking their current occupancy into account. Examples of how this might be approached are identification of a class of motes that are less active, creating less dense partitions, or multiplexing a single LNET over multiple channels.

Uneven distribution and critical motes Critical motes are those that lie in a sparsely populated region and receive a disproportionately large share of traffic due to being responsible for transmitting data for a much larger set of motes. In this case the number of communications channels could be reduced only in the area of concern. This would still require motes in the sparse areas to ferry data from multiple channels across to motes in a more densely populated area, but would result in more robust routes. Motes on the edges of sparsely populated sections or at endpoints may have to communicate over multiple channels.

RAID like redundancy There are several variations of disk RAID (Redundant Array of Independent Disks) that having functionality such as allowing data that is corrupted or missing to be reconstructed using data that exists and parity information. Application of these schemes to our redundant networks may bring further improvements in performance and resiliency.

7.1.2 Extension

Detection of intrusion or failure Since a data sink may receive multiple reports of each event it could compare the various accounts and attempt to detect incorrect

or malicious results. It might also be possible to combine the data from multiple sources to achieve higher resolution data.

Frequency hopping Frequency hopping is used in some protocols to avoid malicious actors from eavesdropping on transmissions. Devices communicating wirelessly agree to switch channels many times per second in an agreed upon order to keep the eavesdropper from being able to easily record the conversation. If mote radio hardware is capable of switching channels rapidly enough without unsustainable power drain then entire LNETs may be able to rotate through a set of channels.

Mobile motes In this dissertation we have made the assumption of static motes but it is not uncommon to think of systems whose constituents are motile. Application of our methods to this type of motescape may require re-engineering the LNET protocol to incorporate continuous integration.

Appendices

Appendix A

Definitions

A.1 Terms

Definitions of selected terms used throughout this text.

Terms

- **Channel** A range of radio frequencies used to convey information between motes using a radio transmitter and a receiver.
- **LNET** A logical network comprised of a subset of motes from M that functions as an independent system.
- **Mote** A tiny and inexpensive computer that contains sensors and uses multiple channel radio for communication.
- **Motescape** The area in which motes are deployed, its characteristics and inhabitants (motes).

A.2 Symbols

Definitions of the symbols used throughout this text.

Symbols

- F A field of motes or *motescape*.
- A The area of F.
- M The set of all motes that populate F.
  - m Some mote $m \in M$.
  - $|M|$ The size of set M, or the number of motes M contains.
- s_0 The radius of a mote's sensor range.
- r_0 The radius of a mote's radio communication range.
- U The unit of area against which density values are calculated.
  - U_1 Base unit of area, $U = 1$.
  - U_s Unit of area equal to sensor coverage area $U = \pi s_0^2$.
  - U_r Unit of area equal to radio coverage area $U = \pi r_0^2$.
- $\deg(m)$ The degree of a mote m or the number of one hop neighbors m has.
- δ Density, the calculated population size per unit of area.
  - $\delta(U)$ Function to calculate density with respect to unit of area U.
  - $\delta_1, \delta_r, \delta_s$ Shorthand for $\delta(U_1)$, $\delta(U_r)$, and $\delta(U_s)$ respectively.

  - $\hat{\delta}$ An approximation of δ.

- μ The mean degree observed in a set of motes.
  - $\mu(S)$ Algorithm for computing the mean degree of motes in set S.
  - μ_r, μ_s Mean degree of devices that have a unit of area equal to U_r, and U_s respectively.

- ℓ Ideal degree, the mote degree that we want motes to have, should maximize throughput and/or minimize collisions.

- C The set of channels (radio frequencies) available for communication. Members of C are represented by consecutive integers starting with 1.
  - c Some channel from the set of channels C where $1 \leq n \leq |C|$.
  - c_{base} Base channel, the default channel for communication and the lowest numbered channel available.

- β Channel capacity, the bandwidth capacity of a single channel.

- γ Capacity factor, the number of channels with capacity β required to satisfy demand.

- φ Redundancy factor, the number of logical redundant networks (LNETs) to divide a motescape into.

A.3 Time periods

These are buffer periods defined to allow motes to synchronize between transitions.

Periods of time

- T_f The initial time that a decision to re-form the network was made.
- W_a Time between neighbor assignments in the distributed turn taker algorithm.
- W_d Start time, or waiting period used to trigger initial setup phase.
- W_e Amount of time to spend evaluating channel suitability.
- W_i Time to wait after the decision to re-form the network before initiating the setup phase.
- W_n Duration of the neighbor discovery period.
- W_o Timeout period before a neighbor unable to communicate with $j\%$ of its neighbors switches to the base channel to begin discovery.
- W_s An optional period for operating in steady state mode after which the setup phase should be triggered.
- W_x Timeout period before switching from channel c to channel $c + 1$ if the discovery process is unable to complete due to external interference.

Bibliography

[1] B. Warneke, M. Scott, B. Leibowitz, L. Z. L. Zhou, C. Bellew, J. Chediak, J. Kahn, B. Boser, and K. Pister, "An autonomous 16 mmˆ3 solar-powered node for distributed wireless sensor networks," *Proceedings of IEEE Sensors*, vol. 2, 2002.

[2] V. Ermolov, M. Heino, A. Karkkainen, R. Lehtiniemi, N. Nefedov, P. Pasanen, Z. Radivojevic, M. Rouvala, T. Ryhanen, E. Seppala, and M. A. Uusitalo, "Significance of Nanotechnology for Future Wireless Devices and Communications," in *2007 IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1–5, IEEE, 2007.

[3] E. A. Lee, "Cyber Physical Systems: Design Challenges," *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pp. 363–369, May 2008.

[4] "NSF Workshop on Cyber-Physical Systems," 2006.

[5] E. Lee, "Cyber-physical systems-are computing foundations adequate," *... Paper for NSF Workshop On Cyber-Physical Systems: ...*, 2006.

[6] V. Hsu, J. Kahn, and K. Pister, "Wireless communications for smart dust," tech. rep., University of California Berkeley, 1998.

[7] H. Lee, A. Cerpa, and P. Levis, "Improving wireless simulation through noise modeling," *Proceedings of the 6th international conference on Information processing in sensor networks - IPSN '07*, p. 21, 2007.

[8] E. A. Lee, P. Asare, D. Broman, M. Torngren, and S. S. Sunder, "CyberPhysicalSystems.org," 2015.

[9] A. a. Cardenas, S. Amin, and S. Sastry, "Secure Control: Towards Survivable Cyber-Physical Systems," *2008 The 28th International Conference on Distributed Computing Systems Workshops*, pp. 495–500, June 2008.

[10] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, pp. 393–422, Mar. 2002.

[11] D. Chen, G. Chang, D. Sun, J. Jia, and X. Wang, "Lightweight key management scheme to enhance the security of internet of things," *International Journal of Wireless and Mobile Computing*, vol. 5, no. 2, p. 191, 2012.

[12] R. Rajkumar, "A CyberPhysical Future," *Proceedings of the IEEE*, pp. 11–14, 2012.

[13] H. Zhou, "Measuring Available Bandwidth for Smart Cyber-Physical Applications," *Tsinghua Science & Technology*, vol. 16, no. 6, pp. 601–610, 2011.

[14] P. Derler, E. a. Lee, and a. S. Vincentelli, "Modeling CyberPhysical Systems," *Proceedings of the IEEE*, vol. 100, pp. 13–28, Jan. 2012.

[15] K. S. Pister, "Smart Dust BAA 97-43," tech. rep.

[16] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems," *Proceedings of the 47th Design Automation Conference on - DAC '10*, p. 731, 2010.

[17] A. Raniwala, K. Gopalan, and T.-c. Chiueh, "Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 8, p. 50, Apr. 2004.

[18] M. Bishop, *Introduction to computer security*. Addison-Wesley Professional, 2004.

[19] B. Pittel, "On spreading a rumor," *SIAM Journal on Applied Mathematics*, vol. 47, no. 1, pp. 213–223, 1987.

[20] N. Fountoulakis, A. Huber, and K. Panagiotou, "Reliable Broadcasting in Random Networks and the Effect of Density," *2010 Proceedings IEEE INFOCOM*, pp. 1–9, Mar. 2010.

[21] A. Frieze and G. Grimmett, "The shortest-path problem for graphs with random arc-lengths," *Discrete Applied Mathematics*, vol. 10, pp. 57–77, 1985.

[22] L. Lazos and R. Poovendran, "Stochastic coverage in heterogeneous sensor networks," *ACM Transactions on Sensor Networks*, vol. 2, pp. 325–358, Aug. 2006.

[23] H. P. Gupta, S. V. Rao, and T. Venkatesh, "Critical Sensor Density for Partial Coverage under Border Effects in Wireless Sensor Networks," *IEEE Transactions on Wireless Communications*, pp. 1–9, 2014.

[24] M. Mansouri, A. Sardouk, L. Merghem-Boulahia, D. Gaiti, H. Snoussi, R. Rahim-Amoud, and C. Richard, "Factors that may influence the performance of wireless sensor networks," in *Smart Wireless Sensor Networks. HD Chinh and YK Tan*, pp. 29–51, http://www.intechopen.com/books/smart-wireless-sensor-networks/factors-that-may-influence-the-performance-of-wireless-sensor-networks, 2010.

[25] L. Kleinrock and J. Silvester, "Optimum transmission radii for packet radio networks or why six is a magic number," *Proceedings of the IEEE National ...*, 1978.

[26] C. Bettstetter, "On the minimum node degree and connectivity of a wireless multihop network," *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing - MobiHoc '02*, p. 80, 2002.

[27] Y. Cheng and T. Robertazzi, "Critical connectivity phenomena in multihop radio models," *..., IEEE Transactions on*, vol. 31, no. 7, pp. 770–777, 1989.

[28] J. Wu and H. Li, "On calculating connected dominating sets for efficient routing in ad hoc wireless networks," in *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*, pp. 7–14, 1999.

[29] J. Kuo, W. Liao, and T. Hou, "Impact of node density on throughput and delay scaling in multi-hop wireless networks," *Wireless Communications, IEEE*, vol. 8, no. 10, pp. 5103–5111, 2009.

[30] C. Efthymiou, S. Nikoletseas, and J. Rolim, "Energy balanced data propagation in wireless sensor networks," *Wireless Networks*, pp. 1–31, 2006.

[31] S. J. Habib, "Modeling and simulating coverage in sensor networks," *Computer Communications*, vol. 30, pp. 1029–1035, Mar. 2007.

[32] X. Wang, G. Xing, Y. Zhang, and C. Lu, "Integrated coverage and connectivity configuration in wireless sensor networks," *... networked sensor ...*, pp. 28–39, 2003.

[33] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak, "Exposure in wireless Ad-Hoc sensor networks," *Proceedings of the 7th annual international conference on Mobile computing and networking - MobiCom '01*, pp. 139–150, 2001.

[34] O. D. Incel, "A survey on multi-channel communication in wireless sensor networks," *Computer Networks*, vol. 55, pp. 3081–3099, Sept. 2011.

[35] R. Vedantham, S. Kakumanu, S. Lakshmanan, and R. Sivakumar, "Component based channel assignment in single radio, multi-channel ad hoc networks," *Proceedings of the 12th annual international conference on Mobile computing and networking - MobiCom '06*, p. 378, 2006.

[36] C.-F. Huang and Y.-C. Tseng, "The Coverage Problem in a Wireless Sensor Network," *Mobile Networks and Applications*, vol. 10, pp. 519–528, Aug. 2005.

[37] H. Zhang and J. C. Hou, "Is Deterministic Deployment Worse than Random Deployment for Wireless Sensor Networks?," *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, no. 4, pp. 1–13, 2006.

[38] J. Singh, C. Holt, and T. Totsuka, "Load Balancing and Data Locality in Adaptive Hierarchical N-Body Methods: Barnes-Hut, Fast Multipole, and Radiosity," *Journal of Parallel and Distributed Computing*, no. 27, pp. 118–141, 1995.

[39] P. Kyasanur and N. Vaidya, "Routing and interface assignment in multi-channel multi-interface wireless networks," in *IEEE Wireless Communications and Networking Conference, 2005*, vol. 4, pp. 2051–2056, IEEE, 2005.

[40] M. W. M. Wang, L. C. L. Ci, P. Z. P. Zhan, and Y. X. Y. Xu, "Multi-channel MAC Protocols in Wireless Ad Hoc and Sensor Networks," *2008 ISECS International Colloquium on Computing Communication Control and Management*, vol. 2, pp. 562–566, 2008.

[41] J. Tang, G. Xue, and W. Zhang, "Interference-aware topology control and QoS routing in multi-channel wireless mesh networks," *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing - MobiHoc '05*, p. 68, 2005.

[42] S. Ivanov, D. Botvich, and S. Balasubramaniam, "Gradient Based Routing Support for Cooperative Multi-channel MAC in Ad Hoc Wireless Networks," *Advanced Information Networking and Applications AINA 2010 24th IEEE International Conference on*, 2010.

[43] J. Bicket, D. Aguayo, S. Biswas, and R. Morris, "Architecture and evaluation of an unplanned 802.11b mesh network," in *Proceedings of the 11th annual international conference on Mobile computing and networking - MobiCom '05*, (New York, New York, USA), p. 31, ACM Press, 2005.

[44] P. Gupta and P. Kumar, "The capacity of wireless networks," *Information Theory, IEEE Transactions on*, vol. 46, pp. 388–404, 2000.

[45] M. Kodialam and T. Nandagopal, "Characterizing the capacity region in multi-radio multi-channel wireless mesh networks," *Proceedings of the 11th annual international conference on Mobile computing and networking - MobiCom '05*, p. 73, 2005.

[46] Z. Li, W.-Z. Li, S. Guo, S.-L. Lu, and D.-X. Chen, "Delay and Capacity Tradeoffs in Mobile Wireless Networks with Infrastructure Support," *Journal of Computer Science and Technology*, vol. 27, pp. 328–340, Mar. 2012.

[47] P. Li, M. Pan, and Y. Fang, "Capacity bounds of three-dimensional wireless ad hoc networks," *Networking, IEEE/ACM Transactions on*, vol. 20, no. 4, pp. 1304–1315, 2012.

[48] P. Kyasanur, N. H. Vaidya, and S. Member, "Capacity of Multichannel Wireless Networks Under the Protocol Model," *IEEE/ACM Transactions on Networking*, vol. 17, no. 2, pp. 515–527, 2009.

[49] Y. Wu, J. a. Stankovic, T. He, and S. Lin, "Realistic and Efficient Multi-Channel Communications in Wireless Sensor Networks," *2008 IEEE INFOCOM - The 27th Conference on Computer Communications*, pp. 1193–1201, Apr. 2008.

[50] M. Neely and E. Modiano, "Capacity and delay tradeoffs for ad hoc mobile networks," *Information Theory, IEEE Transactions ...*, vol. 51, no. 6, pp. 1917–1936, 2005.

[51] R. Cruz and A. Santhanam, "Optimal routing, link scheduling and power control in multihop wireless networks," *... Annual Joint Conference of the IEEE ...*, vol. 00, no. C, 2003.

[52] L. Xiao, S. Boyd, and S. J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributed Computing*, vol. 67, no. 0, pp. 33–46, 2007.

[53] T. Ward, "Robust multihop communication in high density wireless networks," *International Journal of Wireless and Mobile Computing*, vol. 7, no. 5, pp. 428–434, 2014.

[54] R. Ramanathan, "Making ad hoc networks density adaptive," *2001 MILCOM Proceedings Communications for Network-Centric Operations: Creating the Information Force (Cat. No.01CH37277)*, vol. 2, pp. 957–961.

[55] X. Li, H. Shi, and Y. Shang, "A Sorted RSSI Quantization Based Algorithm for Sensor Network Localization," *11th International Conference on Parallel and Distributed Systems (ICPADS'05)*, vol. 1, pp. 557–563, 2005.

[56] Z. Jin, "Improving the performance of distributed simulations of wireless sensor networks," 2010.

[57] A. A. Abbasi and M. Younis, "A survey on clustering algorithms for wireless sensor networks," *Computer Communications*, vol. 30, pp. 2826–2841, Oct. 2007.

[58] Ieee, *IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, vol. 2012. 2012.

[59] S. C. C. S. MAN LAN, "IEEE STD 802.15.4d," p. 317, 2009.

[60] "NS-3 Discrete Event Simulator."

[61] NICTA, "Castalia WSN Simulator."

[62] "TOSSIM TinyOS Simulator."

[63] P. Levis and N. Lee, "Tossim: A simulator for tinyos networks," *UC Berkeley, September*, pp. 1–17, 2003.

[64] "GNU Octave."

[65] E. Jones, T. Oliphant, P. Peterson, and Others, "SciPy: Open source scientific tools for Python," 2001.

[66] "CC2420 2.4 Ghz IEEE 802.15.4 / ZigBee-ready RF Transceiver," Tech. Rep. swrs041, 2006.

[67] D. Moss, J. Hui, P. Levis, and J. Choi, "CC2420 radio stack," *http://www.tinyos.net/tinyos-2.1.0/doc/html/tep126.html*, pp. 1–11, 2007.

[68] D. Moss and J. Hui, "TinyOS Source tos/chips/cc2420/CC2420.h," 2009.

VITA

Theodore Lee Ward

Candidate for the Degree of

Doctor of Philosophy

Thesis: ROBUST NETWORKING IN DENSE WIRELESS NETWORKS

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Doctor of Philosophy in Computer Science at Oklahoma State University, Stillwater, Oklahoma in July, 2015

Completed the requirements for the Master of Science in Computer Science at University of Tulsa, Tulsa, Oklahoma in December, 2005

Completed the requirements for the Bachelor of Science in Computer Science at Oklahoma State University, Stillwater, Oklahoma in December, 1992

Experience:

TransZap Incorporated in Denver, Colorado: Senior Java Developer 2000 to 2002

Independent Software Developer: C++, Java, Visual Basic 1996 to 2000

GeoGraphix Incorporated in Denver, Colorado: C/C++ and VB Software Developer 1993 to 1996

TMS Inc. in Stillwater, Oklahoma: Assembly language and C Software Developer 1993 to 1993

United States Navy Naval Air Station North Island in San Diego, California: Aviation Electronics Technician 1987 to 1990

Professional Memberships:

International Computer Consultants Association: Denver Chapter. 2002

Cyber Security Education Consortium (CSEC)

Association for Computing Machinery (ACM)

Oklahoma InfraGard