UNIVERSITY OF CALIFORNIA, SAN DIEGO

Energy Efficient Strategies for Wireless Sensor Networks with Varying Connectivity Properties

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering (Communication Theory & Systems)

by

Clement Kam

Committee in charge:

Professor William Hodgkiss, Co-Chair
Professor David Sworder, Co-Chair
Professor Robert Bitmead
Professor Kenneth Kreutz-Delgado
Professor Curt Schurgers

2010

The dissertation of Clement Kam is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____
Co-Chair

_____
Co-Chair

University of California, San Diego

2010

DEDICATION

To my wife and family

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

There are a number of people who deserve my gratitude for guiding me through this long journey and helping me see it through to the end. I am first grateful to my main Ph.D. advisor Professor Curt Schurgers, for his direction and generous support during a critical time in my graduate career. His mentorship has been invaluable for my research and personal development, and its impact will last beyond our time at UCSD.

I am also grateful to Professor William Hodgkiss, for his years of guidance, advice, and support, as well as his supervision during the final stages of the Ph.D. process. I also thank Professor David Sworder for his kind suggestions and encouragements. I also thank the members of my thesis committee, Professor Robert Bitmead and Professor Kenneth Kreutz-Delgado, for providing their valuable suggestions and feedback of my constantly evolving research.

There have been a number of friends and colleagues that I have met in San Diego and have meant so much to me. I want to thank my WISL labmates, namely Periklis Liaskovitis, Diba Mirza, and Feng Lu for their inspiration and help around the lab. Thanks to my CEC family, including Simon and Ivy Hui and the Hui family, Jim Yee, Brian and Joy Liu, Eugene and Heather Lee, Tim and Christine Lee, Joe Chang, Steven and May Siu, and so many others for their companionship and support over the years. Also, thanks to the various Harbor Carmel Valley friends whom I met during my final year for their encouragement.

I especially want to recognize my fellow graduate school and associated friends: Alvin AuYoung, Arun Batra, Brendan Morris, Ed and Kia Liao, Allen Chu, Aron Lum, Eric Lin, Marc Krull, Caitlin White, Wayne and Laurie Chen, and others. You have all been an inspiration and encouragement through a very difficult journey, and I would not have made it through without your help. I am a better person for having known you, and the experiences we have shared together will not be forgotten.

Thanks to my special Cornell and New Jersey friends Joseph Chau, Joseph Cheung, Allen Hou, Chris Chen, Robert Aaron, and Amol Rangnekar, who have been there through the various phases of my education and remain close even

though we are physically apart.

I owe so much to my family for their generous love, patience, and support all throughout my life. I would like to thank Kerwin and Priscilla, and Clarence so much for always being there for me. I am so grateful to Mom and Dad for doing so much to raise me right. I hope to continue to make you proud.

Finally, I would like to thank my wife Lulu. Her love, generosity, intellect, and emotional support have provided the fuel and motivation to complete this work. This dissertation is dedicated to you.

Chapter 2 appears, in part, in *Conference Record of the Fortieth Asilomar Conference on Signals, Systems & Computers*. The dissertation author was the primary researcher and author in the publication, and William Hodgkiss supervised the research.

Chapter 3, in part, has been accepted for publication in *IEEE Workshop on Mobile Computing and Emerging Communication Networks (MCECN'10)*. The dissertation author was the primary researcher and author in the publication, and Curt Schurgers supervised the research.

Chapter 4, in part, has been accepted for publication in *IEEE Transactions on Mobile Computing*. The dissertation author was the primary researcher and author in the publication, and Curt Schurgers supervised the research.

VITA

2002	B. S. in Electrical and Computer Engineering, Cornell University
2003-2007	Teaching Assistant, Department of Electrical and Computer Engineering, University of California, San Diego
2004-2010	Research Assistant, Department of Electrical and Computer Engineering, University of California, San Diego
2005	M. S. in Electrical Engineering, University of California, San Diego
2010	Ph. D. in Electrical Engineering, University of California, San Diego

PUBLICATIONS

C. Kam and W. Hodgkiss, "Distributed Target Tracking in a Wireless Sensor Network", *Conference Record of the Fortieth Asilomar Conference on Signals, Systems & Computers*, Pacific Grove, CA, October 2006, pp. 1999–2003.

C. Kam and C. Schurgers, "TreeDMA: a Hybrid MAC/Routing Solution for Small-Scale Wireless Networks," *6th Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services (Mobiquitous 2009)*, Toronto, ON, July 2009, pp. 1–8.

C. Kam and C. Schurgers, "ConverSS: a Hybrid MAC/Routing Solution for Small-Scale, Convergecast Wireless Networks," *3rd International Conference on Signal Processing and Communication Systems (ICSPCS 2009)*, Omaha, NE, Sept. 2009, pp. 1–8.

C. Kam and C. Schurgers, "ConverSS: a Hybrid MAC/Routing Solution for Small-Scale, Convergecast Wireless Networks," to appear in *IEEE Transactions on Mobile Computing*.

C. Kam and C. Schurgers, "Local Information-Based Power Management in a Delay Tolerant Network," to appear in *IEEE Workshop on Mobile Computing and Emerging Communication Networks (MCECN'10)*, Miami, FL, Dec. 2010.

ABSTRACT OF THE DISSERTATION

Energy Efficient Strategies for Wireless Sensor Networks with Varying Connectivity Properties

by

Clement Kam

Doctor of Philosophy in Electrical Engineering (Communication Theory & Systems)

University of California, San Diego, 2010

Professor William Hodgkiss, Co-Chair
Professor David Sworder, Co-Chair

Wireless sensor networks (WSNs) are envisioned as an approach to a wide variety of monitoring applications. From environmental monitoring to military surveillance, the range of scenarios is diverse, as are the constraints on the networking problem. In most cases, these wireless systems are energy-constrained, and it is impractical or infeasible to replace or recharge the batteries. These systems must be designed with the goal of energy efficiency.

In this work, we investigate energy efficient approaches for WSNs that exhibit a variety of connectivity properties. We study three categories of WSNs in

particular. In the traditional view of *multi-hop sensor networks*, there is a large number of sensor nodes with end-to-end connectivity, and they use multi-hop communication. We also consider *dense, small-scale sensor networks*, in which there are only a few nodes, and communication is primarily single-hop. Lastly, we look at *sparse sensor networks*, known as a delay- or disruption-tolerant network (DTN). These networks lack end-to-end connectivity, and must rely on mobility to make connections to route data.

With these three categories of WSNs, we consider approaches that are specifically applied to each unique scenario. For multi-hop WSNs, we use a distributed approach to Kalman filtering for target tracking. By moving the signal processing onto the local nodes, the RF communication, which typically consumes the most power, is reduced. Through simulation, we evaluate the tracking performance.

For our approach to dense, small-scale networks, we exploit the fact that multi-hop communication is rarely needed. We implement a hybrid MAC/routing protocol that assumes that routing is only needed on rare occasions, thus reducing the idle listening energy consumption. For sparse DTNs, we developed a power management scheme that uses the local information available to determine a node's likelihood of delivery, which is used to decide if a node should sleep or listen/beacon. We evaluate these two categories of WSNs through simulation and some experimental work (for the small-scale case), and we show significant gains in energy efficiency over existing approaches.

Chapter 1

Introduction

1.1 Wireless Sensor Networks

Wireless sensor networks have been envisioned as a new method of monitoring the physical world. These systems consist of sensor-equipped wireless devices, capable of performing simple processing tasks. Unlike wired sensing systems, these sensor networks can be deployed in areas that are lacking in infrastructure, providing greater flexibility to conduct a variety of monitoring tasks. Large numbers of these untethered nodes can be deployed for target tracking and surveillance, environmental and habitat monitoring, and healthcare monitoring.

Although these wireless systems can be deployed more readily than wired systems, their usage is constrained by their limited battery capacity. While some systems do rely on energy-harvesting, a large number are still battery-powered. Since it is often impractical or infeasible to recharge or replace these batteries, the operation of the network is tied to a single battery life cycle. Therefore, these systems must be designed with the goal of energy efficiency to maximize the network lifetime.

The vision for sensor networks is typically that of a large number of small, static devices with moderate transmission ranges. A well-known example of such is the ExScal deployment [ARE+05] which was used for intruder detection and classification. Nodes with magnetometer, infrared, and acoustic sensors were placed at the perimeter of a 1.3km×300m area to detect SUV, ATV, or human movement.

Given the coverage perimeter, the typical node transmission range (∼30m), and the number of nodes (10000), the ExScal network had *high connectivity* and required the use of *multi-hop routing*. These are two typical assumptions of systems in sensor network research.

However, other sensor network systems have been proposed that challenge these assumptions of high connectivity and multi-hop routing. For example, networks of sensor-equipped unmanned aerial vehicles (UAVs) have been studied, such as Flying Gridswarms or the indoor UltraSwarm [HWNC05]. These systems consist of a small number of vehicles with transmission ranges on the same order as the coverage area, so multi-hop communication is usually not necessary. However, such systems must still be optimized for energy efficient communication. These vehicles have a limited payload for a battery that powers the propulsion, the sensing subsystem, as well as the communication, so an efficient networking protocol is important.

Other systems that do not conform to the traditional sensor networking paradigm include deployments of few mobile nodes over large areas, such as the 30-node ZebraNet wildlife tracking system [JOW+02] or the proposed Shared Wireless Infostation Model (SWIM) system for gathering biological information of radio-tagged whales [SH03, SH05]. If such systems are to be operational over a long time to complete their study, a shorter range radio would be used, and connectivity would typically be low. These types of networks challenge the assumption of high connectivity that is characteristic of sensor network research.

These real monitoring systems have stretched the traditional view of wireless sensor networks as large, well-connected multi-hop networks of small, static devices. Whether they have fewer nodes or have low connectivity, we must consider energy efficient design approaches that are tailored to these different systems.

1.2 A Taxonomy of Energy Efficient Strategies

Before we discuss the different categories of sensor networks that we study, we first provide an overview of proposed energy efficient strategies that can be

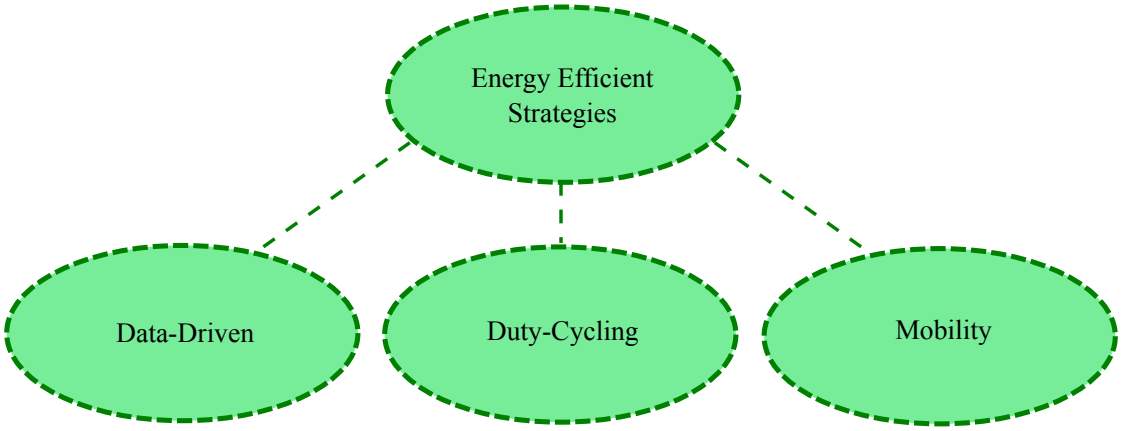


Figure 1.1: Energy conservation schemes.

applied to these systems. The energy conservation strategies that have been used in sensor network research can be categorized as in Fig. 1.1, adopting the taxonomy used in the Anastasi survey paper [ACFP09]. *Data-driven* approaches involve reducing the amount of sensor data to be communicated over the network, since communication typically consumes significantly more energy than data processing [RSPS02]. On the other hand, *duty-cycling* approaches involve controlling the radio transceiver to be either awake, for sending and receiving, or in the low-power sleep state, when no communication is needed. In the following subsections, we delve deeper into these two categories. *Mobility* approaches introduce additional mobile entities into the network to collect data. We do not consider using any entities outside of the main network, and thus do not consider such approaches in our work.

1.2.1 Data-Driven Approaches

The first category of energy efficient strategies we consider is data-driven techniques. It has been shown that in a typical sensor network, computation usually consumes about 3 orders of magnitude less energy than communication [PK00]. The goal of data-driven approaches is to reduce the amount of data samples transmitted through the network, while keeping the sensing accuracy within acceptable levels for the particular application. One of the most studied data-driven methods

is *in-network processing* or *data aggregation*. The idea is to move the computation away from a remote processor and onto the nodes local to the sensed phenomena, thus trading off communication for local computation.

For example, many deployed sensing systems continuously record and report the unprocessed data to a remote site, as was done in the Volcán Reventador sensor network [WALW+06]. However, by doing some processing, combining, or filtering of the raw data locally on the nodes, the amount of data that is communicated to the remote site can be reduced, along with the consumed energy. To achieve a greater degree of in-network processing, data is not routed according to a shortest path metric, but instead is routed through points where more data aggregation can occur. This technique is known as *data-centric routing* [FRWZ00].

1.2.2 Duty-Cycling Approaches

The other major category of energy conservation approaches we consider is duty-cycling. Duty-cycling involves implementing a sleep/wakeup scheme to reduce radio energy consumption. Duty-cycling breaks down further into two broad categories: *topology control* and *power management*.

When there are sufficiently many nodes, as in a typical sensor network, there is usually redundant coverage. In topology control, this node redundancy is exploited such that only a select subset of nodes are awake at a time. Node redundancy can be viewed in terms of either geographic location or network connectivity. Location-driven topology control schemes cycle through nodes that are in the same geographic area and have redundant sensor information [XHE01, ZR03]. Connectivity-driven topology control schemes select the subset of nodes that provides sufficient sensing coverage and still allows for a connected network [CJBM02, CE04].

In power management schemes, nodes independently implement a sleep/ wakeup cycle. Power management can be viewed as a complementary technique to topology control, such that the subset of nodes that are covering the area control the radio sleep/awake state at a finer granularity. Power management can be implemented in different layers of the network architecture. Some wakeup schemes

run on top of a medium access control (MAC) protocol [STS02, STGS02]. This provides greater flexibility since they are agnostic regarding what MAC is used. Other wakeup schemes are implemented directly in a MAC [YHE04, vDL03], which can enable superior performance due to tighter integration.

1.3 Sensor Network Classification

In this section, we describe how we classify the sensor network systems that we study, and for each class we describe which approaches to energy efficiency are most applicable. To summarize the differences between types of networks, we present a classification using our own nomenclature. The two network parameters that we consider varying are the number of nodes N and the relative transmission range $r/\sqrt{A}$, where r is the transmission range, and A is the deployment area. We normalize the transmission range by $\sqrt{A}$, because scaling both r and $\sqrt{A}$ identically will have no effect on the network connectivity properties. A diagram of our network classification is shown in Fig. 1.2.

1.3.1 Multi-Hop WSNs

For the first case we consider, we have r is small relative to $\sqrt{A}$ and N is sufficiently large such that the entire network is connected. We call this class of networks *Multi-Hop WSNs* (Fig. 1.2). Because the transmission range is small relative to the deployment area, nodes at different locations in the network are typically unable to communicate directly. Instead, intermediate nodes are required to relay data, forming multi-hop communication routes. These multi-hop topologies are commonly assumed in sensor network research, as seen in the ExScal deployment. Sensor networks are usually envisioned as collections of small devices with relatively short range radios, and the number of nodes in the network is large enough such all of the nodes are connected.

Most energy conservation approaches are designed for this range of WSNs. Both duty-cycling and data-driven approaches apply. As an example, our work presents a WSN application that works within the framework of a specific energy

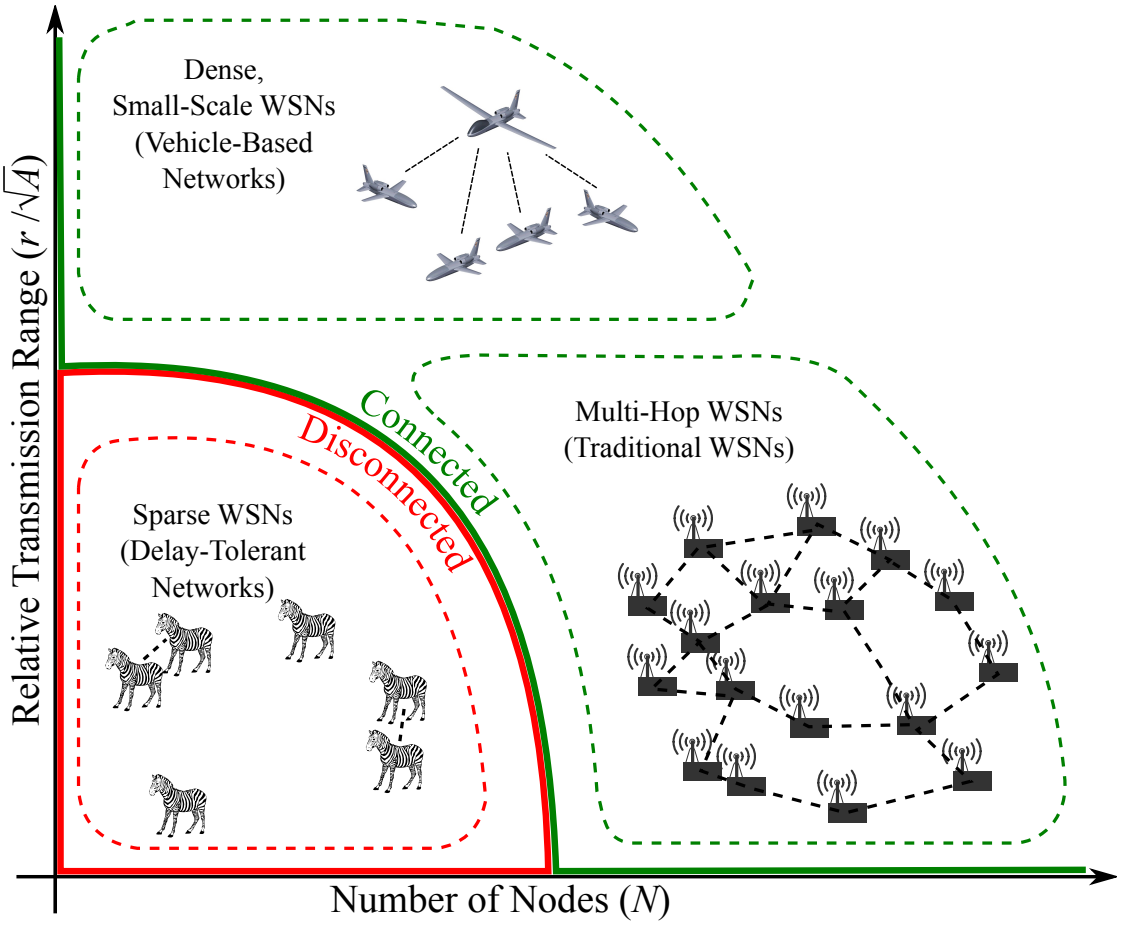


Figure 1.2: Wireless sensor network classification.

conservation approach. This application is a target tracking system consisting of a large number of static sensors. In traditional target tracking systems, there are a small number of sensors, and the sensor signals are processed together to get an up-to-date location of the target. Our challenge is to adapt this approach to a system employing many energy-constrained sensor nodes. The energy efficient strategy that handles this challenge most directly is a data-driven in-network processing approach. In this problem, we assume a distributed computing network architecture, in which the target tracking algorithm is computed on nodes local to the detected event. This approach reduces the amount of energy-draining communication as compared to a centralized approach, in which all of the raw data is transmitted to a remote site for processing.

1.3.2 Sparse WSNs

To transition to the second class of networks, we take the previous case of multi-hop WSNs but consider removing nodes from the network. In effect, r remains small relative to $\sqrt{A}$, but we reduce N until the network has disconnected portions. We term this class of networks *Sparse WSNs* (Fig. 1.2). With such low connectivity and relatively small transmission ranges, communication and networking is severely impaired. However, if mobility is introduced into the system, nodes can physically move to come in radio range of each another, thus enabling data routing. These kinds of mobile networks are known as *delay-* or *disruption-tolerant networks* (DTNs). We typically encounter these types of networks in some wildlife tracking systems, provided N and r are relatively small.

Because of certain characteristics of DTNs, there are limited options for applying data-driven energy conservation schemes. Data-driven approaches, particularly in-network processing, rely on using data-centric metrics for routing to enable more data aggregation. However, many DTNs do not assume control over node mobility, so data messages are handed off upon contact with another node.

Duty-cycling with topology control is also not very applicable to sparse WSNs. Topology control exploits redundancy in the network and can cycle through nodes in a coordinated manner. For sparse DTNs, the coverage is not enough to

provide sufficient redundancy to cycle through nodes.

The most suitable strategy for DTNs is duty-cycling with power management, since these are simply sleep/wakeup cycles that depend on the application. For DTNs, the sensible approach is to design the power management to run on top of the MAC. The typical MAC for DTNs involves some sort of beaconing mechanism for neighbor discovery, since nodes are typically not in contact. Our approach is to design a power management that runs on top of this baseline beaconing.

1.3.3 Dense, Small-Scale WSNs

For the last class of networks, we consider a sparse WSN and bring the nodes closer together. This has the effect of shrinking the coverage area A, so that the relative transmission range $r/\sqrt{A}$ increases. Eventually, nodes are brought in close enough proximity that the network becomes connected again. We call this class of networks *Dense, Small-Scale WSNs* (Fig. 1.2). Because the number of nodes is small, connectivity is only achieved when nodes are so close together that the network has near full connectivity (each pair of nodes can communicate directly with one another). The characteristic of these networks is that communication routes between nodes are mostly one or two hops in length.

The application that we are likely to find this class of WSNs is a vehicle-based sensor network, such as swarming UAV systems. The transmission range of these systems are on the same scale as the deployment area, since there are a small number of vehicles that must communicate with each other on a regular basis.

For dense, small-scale WSNs, data-driven approaches for energy conservation do not provide a significant gain in efficiency. Any gain from in-network processing will be severely limited because most nodes are in communication range of each other. The distance from nodes to the sink will only be one or two hops, so it is simple enough to route the unprocessed data to the sink.

Duty-cycling approaches with topology control are also not very applicable to these networks. The number of nodes is so low for these small-scale networks, so the node redundancy would usually be insufficient for topology control.

As with sparse WSNs, the most suitable approach for dense, small-scale

WSNs is duty-cycling with power management. Although running a power management on top of the MAC would work, having the power management integrated into the MAC turns out to be much more efficient. Because the network is simply a small number of nodes in close range, the MAC function is most important. Therefore, a MAC with a built-in wakeup schedule can provide greater energy savings.

1.4 Contributions and Overview

This dissertation studies specific, novel energy efficient strategies for the three classes of networks presented in the previous section. In Chapter 2, we study the multi-hop WSN target tracking problem. In accordance with the in-network processing approach, we implement a Kalman filter within a distributed computing network architecture. The network is composed of a field of static proximity sensors, which are triggered by a single moving target. The task is to take the series of triggered sensor events, which are spaced irregularly in time, and accurately track the target location. We develop our approach by starting with a centralized tracking scenario, and we gradually adapt it to the proximity sensor case with intermittent sampling.

Chapter 3 deals with the sparse delay-tolerant network. Many DTN applications are rich with information that is local to each node, such as location, velocity, or history of contacts. Such information is typically used for delay-tolerant routing to reduce delivery delay or improve delivery success. Our novel approach to power management in DTNs is to use such local state information in a sleep/wakeup scheme. We identify geographic convergecast as a particular application that is rich in this kind of information. For this application, we develop a sleep/wakeup decision that is a function of the local state. This is shown to outperform other power management approaches that do not fully leverage the available information.

In Chapter 4, we handle the dense, small-scale swarming UAV network. To our knowledge, no solutions have been proposed that specifically optimize the

communication for these kinds of networks. The application we study is a continuous monitoring system, in which nodes report all the data to the sink. This type of traffic pattern is also known as convergecast. Our approach to energy efficiency is to implement a combined MAC/routing duty-cycling protocol, which we show to be more efficient than having separate routing and MAC layers.

In Chapter 5, we provide conclusions and future directions for this work.

1.5 Impact

The main impact of our work is that it extends the possibilities for conducting monitoring studies for different sensing systems. For example, without energy efficient solutions, missions for autonomous vehicle networks would have stricter time constraints to perform their tasks. As another example, a DTN monitoring system would have a shorter overall lifetime, either cutting the study short or requiring more frequent re-deployment. By tailoring our approaches to the specific classes of networks, we can provide energy efficient solutions with performance that exceeds that of traditional WSN solutions. Our work enables a variety of sensing systems to extend the operation for long-term study and for greater flexibility at the application level.

Chapter 2

Multi-Hop WSNs: A Data-Driven Approach

In this chapter, we study an example of the traditional view of multi-hop WSNs. We consider the task of designing a target tracking algorithm for an energy-constrained multi-hop WSN deployed over a large area. The assumption in the target tracking problem is that the network is operating within a data-driven, in-network processing framework. With this assumption, we first describe the generic target tracking algorithm that is traditionally used, and then we demonstrate how to adapt it within our distributed sensor network framework.

2.1 Introduction

Target tracking is an essential task for commercial and military applications. For example, traffic monitoring and facility security are applications that require the ability to track an object[LRZ03]. Energy-aware target tracking for wireless sensor networks is recognized as an important and challenging problem [BRS03]. In this problem, sensor information from multiple nodes must be processed together to accomplish the tracking task. One approach to processing the sensor data is to do so within a *centralized* architecture, in which all of the raw sensor measurements are sent to a single remote site to be processed. This architecture, however, demands significant bandwidth for communication of all the sensor information, which can

put a strain on the sensors' on-board batteries [CZMK03]. Other issues with a centralized architecture are high latency due to the increased traffic, and the lack of robustness with the central site being a single point of failure.

To reduce the communication burden, a *distributed* architecture is preferred for wireless sensor networks. In a distributed architecture, a different node is chosen locally to be a processing node, and the chosen node changes with the target position. A subset of local sensors in the network is chosen to collect sensor data while the remaining sensors are placed in a power-saving sleep mode. An added advantage to using a distributed architecture is its robustness to network changes and node failures [CK03].

In this chapter, we look at an example of a distributed tracking algorithm for a moving target. We assume that the target has been detected prior to track initiation. At each sensor event, all of the data processing occurs at the triggered node. We consider the problem of tracking only a single target. Extension to the multiple target tracking problem is not considered.

In the rest of the chapter we set up the basic Kalman filtering framework that will be used for target tracking. Then we describe the positioning of the nodes in the sensor network and how the proximity sensor data is used as input in the filtering framework. We then develop our approach to adapt the Kalman filter to the distributed sensor network. Finally, we evaluate our approach though simulation and provide our insights.

2.2 Kalman Filtering

The Kalman filter is a plant-model based estimator that minimizes the mean squared error at each iteration of the algorithm [BSLK01]. The Kalman filter is a common approach used for target tracking applications. The state-space representation of the target's state dynamics and the observation are provided in this section. The Kalman filter equations, which estimate the state, also are given.

State Dynamics Equation

For tracking applications, the state vector $\mathbf{x}_n$ typically includes the target position, velocity, and any other additional dynamics or properties of the target. The time evolution of the target state due to the dynamics is given by the following equation:

$$\mathbf{x}_{n+1} = \mathbf{\Phi}\mathbf{x}_n + \mathbf{\Gamma}\mathbf{w}_n \tag{2.1}$$

where $\mathbf{\Phi}$ is the state transition matrix, $\mathbf{\Gamma}$ is the process noise matrix, and $\mathbf{w}_n$ is the Gaussian process noise vector with distribution $\mathcal{N}(\mathbf{0}, \mathbf{Q})$.

Observation Equation

The observation vector $\mathbf{y}_n$ is the recorded data as a function of the state vector. This vector is used as input into the Kalman filter. The observation equation is given by

$$\mathbf{y}_n = \mathbf{H}\mathbf{x}_n + \mathbf{n}_n \tag{2.2}$$

where $\mathbf{H}$ is the observation matrix, and $\mathbf{n}_n$ is the Gaussian observation noise vector with probability distribution $\mathcal{N}(\mathbf{0}, \mathbf{R})$.

Filtering Equations

The Kalman filter uses the state dynamics plant model and filters the input observations $\mathbf{y}_n$ to estimate the state vector. It operates in two phases: a prediction phase and a correction phase. The prediction equations predict the next state as follows:

$$\boldsymbol{x}_{n+1}^{-} = \mathbf{\Phi}\hat{\mathbf{x}}_n \tag{2.3}$$

$$\boldsymbol{P}_{n+1}^{-} = \mathbf{\Phi}\mathbf{P}_n\mathbf{\Phi}^T + \mathbf{\Gamma}\boldsymbol{Q}\mathbf{\Gamma}^T \tag{2.4}$$

where $\mathbf{P}_n^-$ is the matrix that describes the covariance of the error between the predicted and actual state vector.

The correction equations then incorporate the current observation $\mathbf{y}_{n+1}$ to

correct the prediction:

$$\mathbf{K}_{n+1} = \mathbf{P}_{n+1}^{-}\mathbf{H}^T(\mathbf{H}\mathbf{P}_{n+1}^{-}\mathbf{H}^T + \mathbf{R})^{-1} \quad (2.5)$$

$$\hat{\mathbf{x}}_{n+1} = \mathbf{x}_{n+1}^{-} + \mathbf{K}_{n+1}(\mathbf{y}_{n+1} - \mathbf{H}\mathbf{x}_{n+1}^{-}) \quad (2.6)$$

$$\mathbf{P}_{n+1} = (\mathbf{I} - \mathbf{K}_{n+1}\mathbf{H})\mathbf{P}_{n+1}^{-} \quad (2.7)$$

where $\mathbf{K}_n$ is the Kalman gain, $\hat{\mathbf{x}}_n$ is the state estimate, and $\mathbf{P}_n$ is the covariance of the error between the estimated (corrected) and actual state vector.

2.3 Target Dynamics

In this section, we provide the equations that define the dynamics for our particular target motion model. The state vector is given by

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{pmatrix} \quad (2.8)$$

whose elements are x position, y position, x velocity, and y velocity. At the start of data collection, the initial state vector is given by

$$\mathbf{x}_0 = \begin{pmatrix} x_{init} \\ y_{init} \\ \dot{x}_{init} \\ \dot{y}_{init} \end{pmatrix} \quad (2.9)$$

The target motion model used in this work is the (nearly) constant velocity (CV) model [BSLK01], whose state transition matrix and process noise matrix are

given by

$$\mathbf{\Phi} = \begin{pmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.10}$$

$$\mathbf{\Gamma} = \begin{pmatrix} \frac{T^2}{2} & 0 \\ 0 & \frac{T^2}{2} \\ T & 0 \\ 0 & T \end{pmatrix} \tag{2.11}$$

The process noise $\mathbf{w}_n$ is an independent Gaussian random process with covariance matrix

$$\mathbf{Q} = \begin{pmatrix} qT & 0 \\ 0 & qT \end{pmatrix} \tag{2.12}$$

where $q = 1$ and $T = 0.1$ is the time interval between samples.

2.4 Sensor Field Setup

The sensor field is generated in a manner similar to the work by McErlean and Narayanan [MN02] and is set up as follows:

- 900 nodes are placed in a random grid-like formation, initially centered at grid points spaced 30 meters apart: $(x, y) = (30i, 30j)$ (meters), where $i = 1, 2, \ldots, 30$, and $j = 1, 2, \ldots, 30$.

- Actual node locations deviate from the grid points by additive Gaussian random variables with a standard deviation of $\sigma_{pos} = 2$m in both x and y coordinates. The locations are stored in a matrix:

$$\mathbf{A} = \begin{pmatrix} x_A(1) & y_A(1) \\ x_A(2) & y_A(2) \\ \vdots & \vdots \\ x_A(900) & y_A(900) \end{pmatrix} \tag{2.13}$$

  An example of a sensor field is shown in Fig. 2.1.

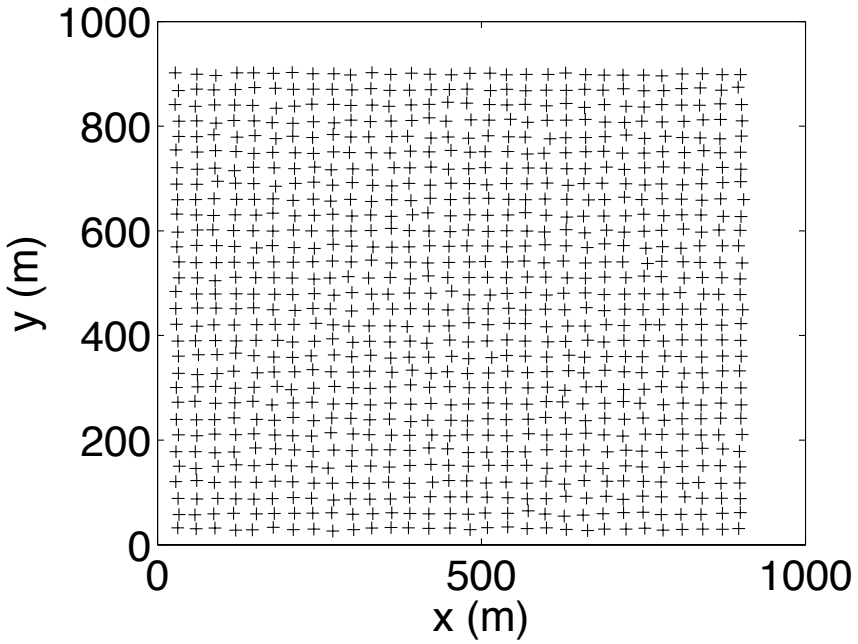


Figure 2.1: A sample deployment of a field of sensor nodes.

- The node locations are assumed to contain some associated error. These assumed node locations deviate from the actual node locations by additive Gaussian random variables with a standard deviation of $\sigma_b = 0.4$m in both x and y coordinates. The locations are stored in a matrix:

$$\mathbf{L} = \begin{pmatrix} x_L(1) & y_L(1) \\ x_L(2) & y_L(2) \\ \vdots & \vdots \\ x_L(900) & y_L(900) \end{pmatrix} \tag{2.14}$$

  These assumed locations are stored in each sensor and used as inputs to the tracking algorithm.

2.5 Proximity Sensor Data

The type of sensor that we consider in this work is a simple proximity sensor, which only detects whether a target is or is not within a sensing range d.

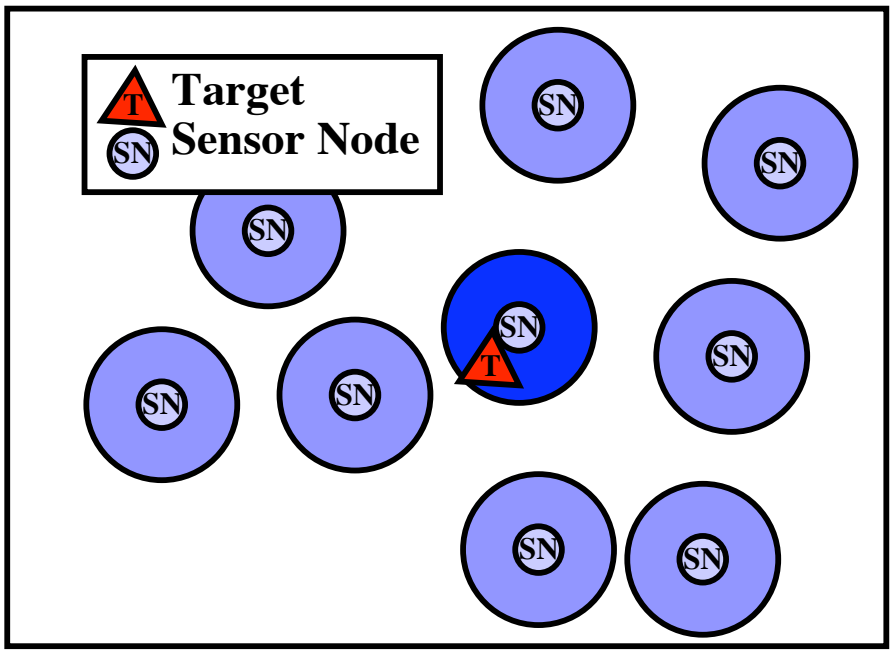


Figure 2.2: Proximity sensor model.

This type of sensor model is illustrated in Fig. 2.2. The measurement $\mathbf{y1}_n$ is the assumed location of the triggered sensor node. We assume the time associated with the measurement (n) is halfway between the time the target enters and exits the sensor's range. Unlike a traditional tracking system, these time measurements do not occur at regular intervals. This is due to having multiple proximity sensors that are dispersed over a large area. To adapt the Kalman filter to this scenario of sparsely sampled data, we first study the basic Kalman filtering case with regularly sampled measurements. Then we look at a case where we remove some of the measurements such that the remaining measurements are irregularly distributed in time, and we adapt the Kalman filter for this case. Finally, we can apply the same principles to adapt the Kalman filter to the proximity sensor case.

2.5.1 Tracking With Noisy Position Measurements

Before simulating the actual proximity sensor network, we first simulate a simple two-dimensional tracking scenario in which noisy position measurements are made at regular time instants. The target motion is the same as in Section 2.3,

and the initial position of the target is given by $x_{init} = 5 + n_x$ and $y_{init} = 5 + n_y$, where n_x and n_y are independently and identically distributed as $\mathcal{N}(0, \sigma^2_{init})$. The initial velocity of the target is given by $\dot{x}_{init}$ and $\dot{y}_{init}$, both uniform randomly distributed over the interval $[0, v_{max}]$ (we choose $v_{max} = 10$). The noisy position is reported as $\mathbf{y0}_n$ at the n^{th} detection according to

$$\mathbf{y0}_n = \mathbf{H}\mathbf{x}_n + \mathbf{n}_n \tag{2.15}$$

where $\mathbf{n}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ is the measurement noise at time sample n, and

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{2.16}$$

$$\mathbf{R} = \begin{pmatrix} \sigma^2_{meas} & 0 \\ 0 & \sigma^2_{meas} \end{pmatrix} \tag{2.17}$$

where here we choose $\sigma^2_{meas} = \sigma^2_{init} = 2.25$. The Kalman filter starts with the initial state:

$$\hat{\mathbf{x}}_0 = \begin{pmatrix} E[x_{init}] \\ E[y_{init}] \\ E[\dot{x}_{init}] \\ E[\dot{y}_{init}] \end{pmatrix} = \begin{pmatrix} 5 \\ 5 \\ 5 \\ 5 \end{pmatrix} \tag{2.18}$$

$$\mathbf{P}_0 = \begin{pmatrix} \sigma^2_{init} & 0 & 0 & 0 \\ 0 & \sigma^2_{init} & 0 & 0 \\ 0 & 0 & \frac{v^2_{max}}{12} & 0 \\ 0 & 0 & 0 & \frac{v^2_{max}}{12} \end{pmatrix} \tag{2.19}$$

where the σ^2_{init} terms come from the initial uncertainty in starting location, and the $\frac{v^2_{max}}{12}$ terms are calculated by taking the second moment of the uniformly distributed initial velocity. The Kalman filter uses the parameters from the constant velocity motion model (Section 2.3), and it filters the $\mathbf{y0}_n$ measurements to get estimates of the target state $\hat{\mathbf{x}}_n$, which contains the target position. A simulated example of this tracking scenario is shown in Fig. 2.3. The target moves from the bottom-left of the figure to the top-right. From the figure, we observe that the filtered data

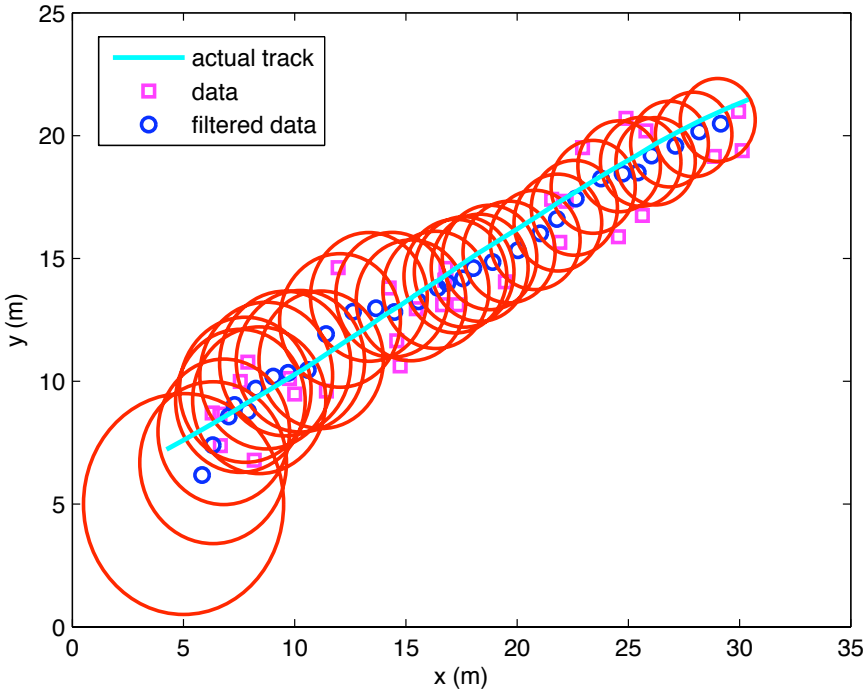


Figure 2.3: Target tracking for noisy position measurements, with 3σ prediction circles.

improves on the noisy measurement. The circles show where the Kalman filter predicts the target should be within a 3σ radius. These circles are defined by the predicted position (from $\mathbf{x}_{n+1}^{-}$) and predicted position error variance (from $\mathbf{P}_{n+1}^{-}$). The predicted position error variance in the x (or equivalently, the y) coordinate is shown in Fig. 2.4. Fig. 2.4 shows that the error variance of the predicted x (or y) position decreases and flattens out with more data, indicating an increasing level of certainty in our estimates.

2.5.2 Tracking With Sparse Noisy Position Measurements

As an intermediate step between the tracking problem with noisy position measurements and the proximity sensor problem, we study the scenario of tracking when data samples are missing. A distinction between the proximity sensor network scenario and the generic two dimensional tracking problem is that in the

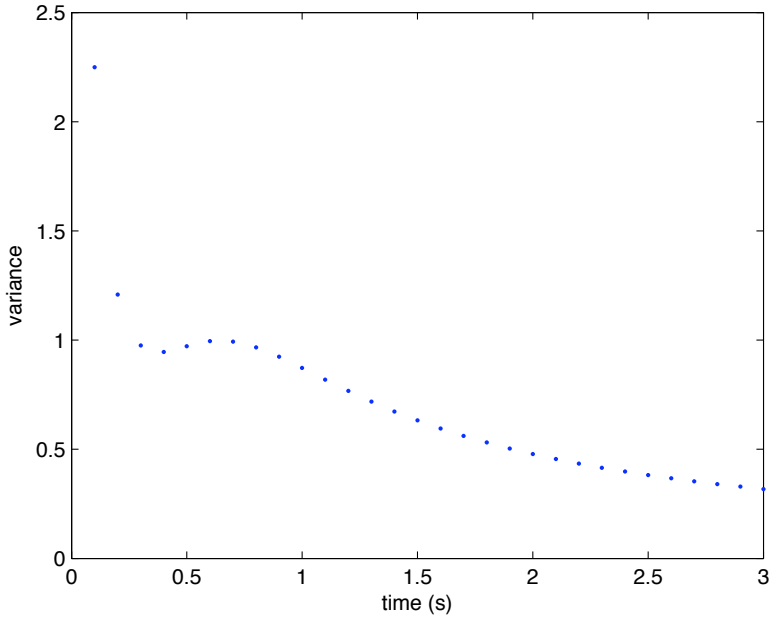


Figure 2.4: Noisy position measurements predicted position error variance $\mathbf{P}_n^-$ in the x (or equivalently, y) direction.

proximity sensor network, the data is not available at a constant sampling rate. To study the performance of the Kalman filter when the samples are spaced irregularly in time, we repeat the tracking simulation of Subsection 2.5.1, but we remove a large section of the measurement data. For the samples where no measurement data is available, only the prediction step (Eqns. (2.3)-(2.4)) of the Kalman filter is carried out, and $\hat{\mathbf{x}}_{n+1}$ and $\mathbf{P}_{n+1}$ are set to $\mathbf{x}_{n+1}^-$ and $\mathbf{P}_{n+1}^-$, respectively.

Fig. 2.5 shows the result of the simulation. Again, the target moves from the bottom-left of the figure to the top-right. The solid line indicates the samples for which measurements are available, and the dotted line indicates when no measurements are available. The large ellipses are the 3σ prediction error ellipses. After the 10 samples without data, the prediction for the target location is less certain, and resulting in a higher error variance, and a larger 3σ prediction error ellipse. This larger ellipse includes the beginning of the connected line segment near position (22m,24m), where the measurements resume. This shows that although no measurements of the target are available for those 10 samples, the prediction

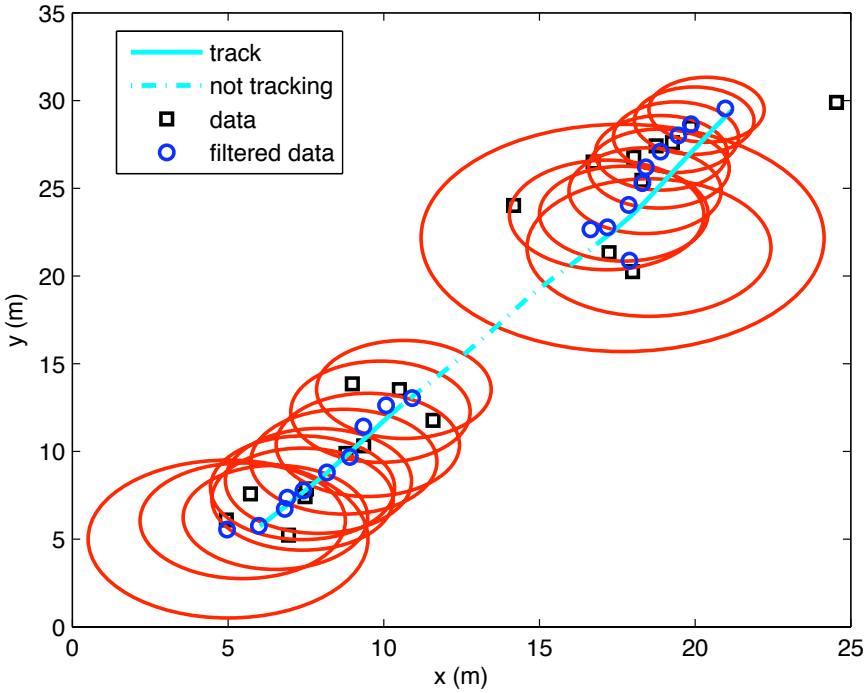


Figure 2.5: Target tracking for sparse position measurements with 3σ prediction circles.

error ellipse is still able to encompass the target's position when the measurements are available again. The predicted error variance is plotted in Fig. 2.6. We observe that the variance (uncertainty) of the prediction in the x or y coordinate decreases when data is present but grows sharply in the absence of data.

2.5.3 Tracking With Proximity Sensor Measurements

We now study the proximity sensor network scenario, in which a sensor node is triggered if a target is present within the sensing range d (chosen here to be 15m). When a sensor is triggered, it reports its location as the measurement. The time that corresponds to this measurement is halfway into the time interval over which the target is in sensing range. These measurements occur very sparsely in time. We adopt the approach from Subsection 2.5.2, so that when no measurement is available, only the prediction step of the Kalman Filter is carried out. The assumed

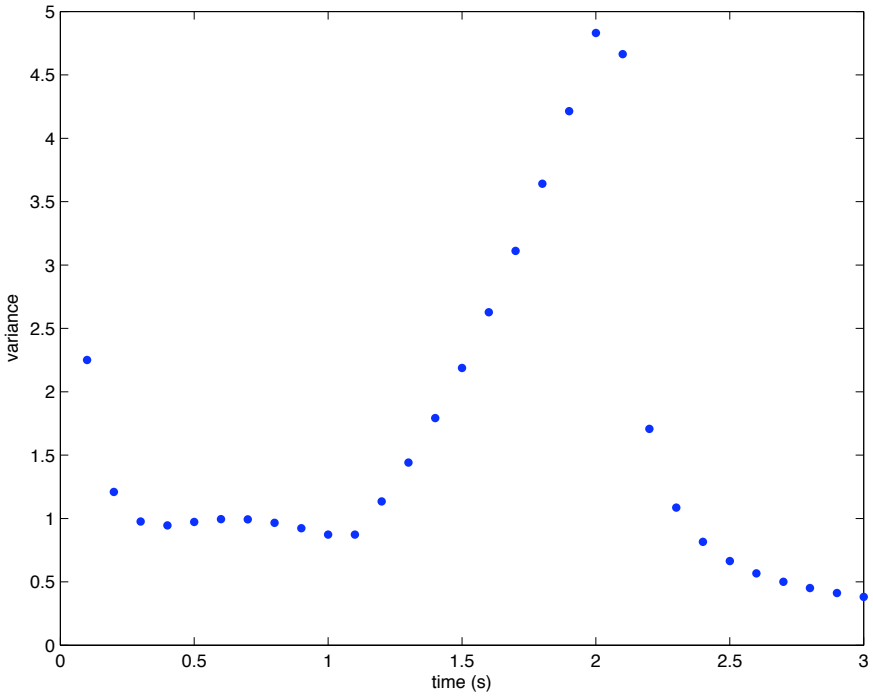


Figure 2.6: Sparse measurements predicted position error variance $\mathbf{P}_n^-$ in the x (or equivalently, y) direction.

sensor location $\mathbf{L}_j$ is reported at the n^{th} time sample as

$$\mathbf{y1}_n = \begin{pmatrix} L_{j,1} \\ L_{j,2} \end{pmatrix} \tag{2.20}$$

To adapt these measurements to the state space observation equation (Eqn. (2.2)), we model the Gaussian noise term $\mathbf{n}_n$ as having two sources of noise: (1) the difference between the actual triggered node location ($\mathbf{A}_j$) and the assumed node location ($\mathbf{L}_j$) and (2) the difference between the target location ($\mathbf{Hx}_n$) and the actual triggered node location ($\mathbf{A}_j$). The resulting error covariance matrix is given by

$$\mathbf{R} = \begin{pmatrix} \frac{d^2}{3} + \sigma_b^2 & 0 \\ 0 & \frac{d^2}{3} + \sigma_b^2 \end{pmatrix} \tag{2.21}$$

The σ_b^2 term is from the uncertainty in the assumed node locations. The $\frac{d^2}{3}$ term comes from target location error, in which we using the second moment of a uniform distribution to approximate the Gaussian term. The target location is modeled as uniformly distributed in the jth sensing range $[A_{j,k} - d, A_{j,k} + d]$ in both the x and y coordinates.

The Kalman filter is initiated at the first detection, so the initial state estimate is the corresponding assumed sensor location, or

$$\hat{\mathbf{x}}_0 = \begin{pmatrix} E[x_{init}] \\ E[y_{init}] \\ E[\dot{x}_{init}] \\ E[\dot{y}_{init}] \end{pmatrix} = \begin{pmatrix} y1_{0,1} \\ y1_{0,2} \\ 5 \\ 5 \end{pmatrix} \tag{2.22}$$

$$\mathbf{P}_0 = \begin{pmatrix} \frac{d^2}{3} + \sigma_b^2 & 0 & 0 & 0 \\ 0 & \frac{d^2}{3} + \sigma_b^2 & 0 & 0 \\ 0 & 0 & \frac{v_{max}^2}{12} & 0 \\ 0 & 0 & 0 & \frac{v_{max}^2}{12} \end{pmatrix} \tag{2.23}$$

where the $\frac{v_{max}^2}{12}$ terms are calculated by taking the second moment of the uniformly distributed initial velocity.

2.6 Simulation Evaluation

We simulated a target moving within a sensor field using the motion dynamics given in Section 2.3, and we assessed the performance of each tracking system using the mean-squared error between the true and estimated positions. The simulation is run for 1000 samples or until the target is outside of the sensor field:

$$x_{n,1}, x_{n,2} \notin [30, 900] \tag{2.24}$$

Fig. 2.7 is one realization of the proximity sensor simulation of the target moving through the sensor network. We observe that the track estimates are able to approximate the actual target path better than the sensor locations, demonstrating the work of Kalman filter. Fig. 2.8 is a zoomed-in version of the proximity sensor simulation with the 3σ circles shown, which, with high probability, encompass the target position at the times of detection. These 3σ circles can be used to predict the location of the target so that neighboring nodes can be alerted of the target's arrival. They can also be used for electing the next local processor node.

Fig. 2.9 shows the error variance of the predicted position as a function of time. As we have observed with the simulation from Section 2.5.2, the error variance increases sharply in the absence of measurement data, when only the prediction step of the Kalman filter is carried out. When a measurement is reported, the error variance decreases drastically. As time progresses and samples are provided, the general trend is that the error variance decreases, indicating an increase in confidence in our track estimates. Fig. 2.10 plots the squared error in the position as a function of time. We can see that while the error of the unfiltered positions does not display a trend, the error of the filtered positions shows a downward trend with time. In other words, while the observations have the same error variance over time, the error variance of the estimates trends downward.

In Figs. 2.11-2.14, we also show a target whose motion dynamics have a slightly curved track, and is therefore not exactly matched to the CV motion model of the Kalman filter. The position error still has a downward trend, and the error ellipses can still encompass the target position, though not quite as well as if the motion models were matched. If the mismatch was worse, one simple

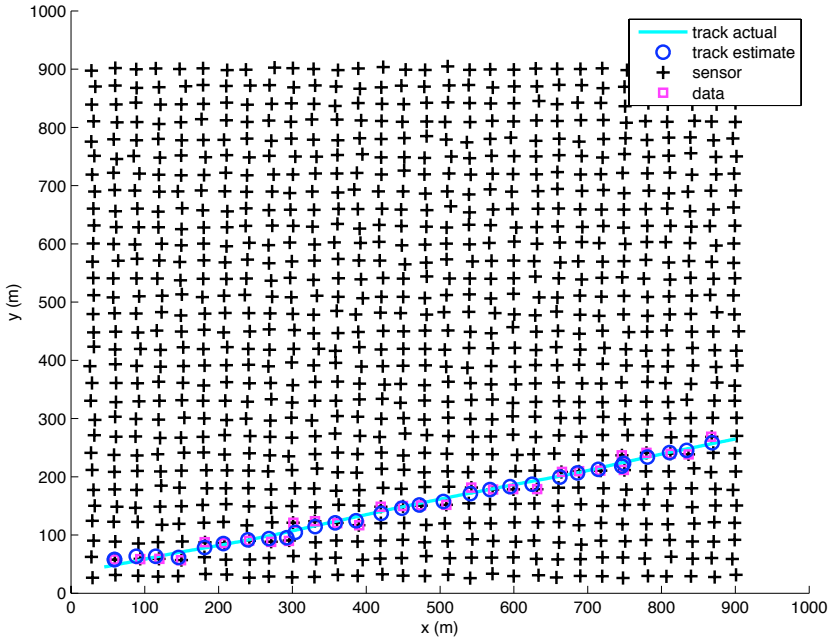


Figure 2.7: Case 1: Simulation of target tracking with proximity sensors, $d = 15$, $q = 1$.

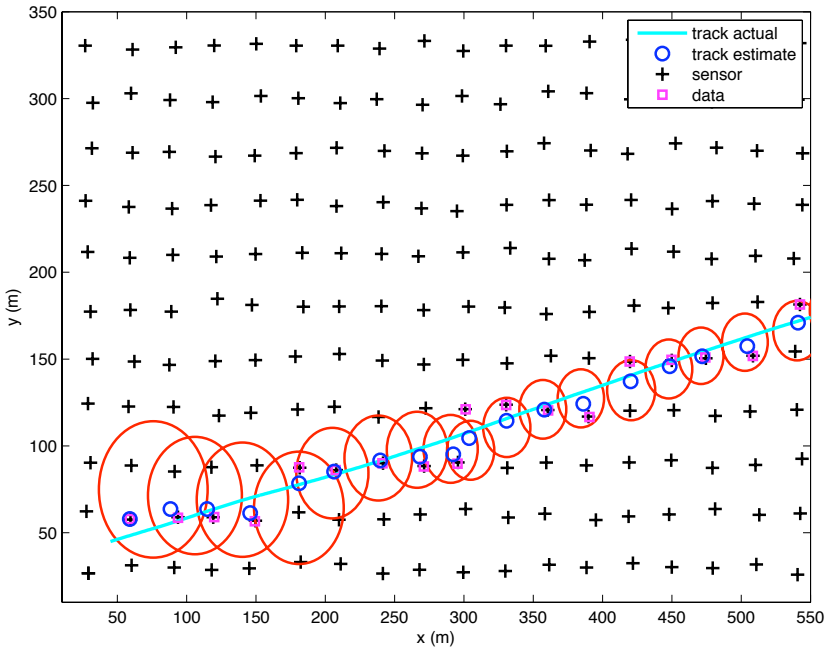


Figure 2.8: Case 1: Simulation of target tracking with proximity sensors (zoomed version), with 3σ prediction circles, $d = 15$, $q = 1$.

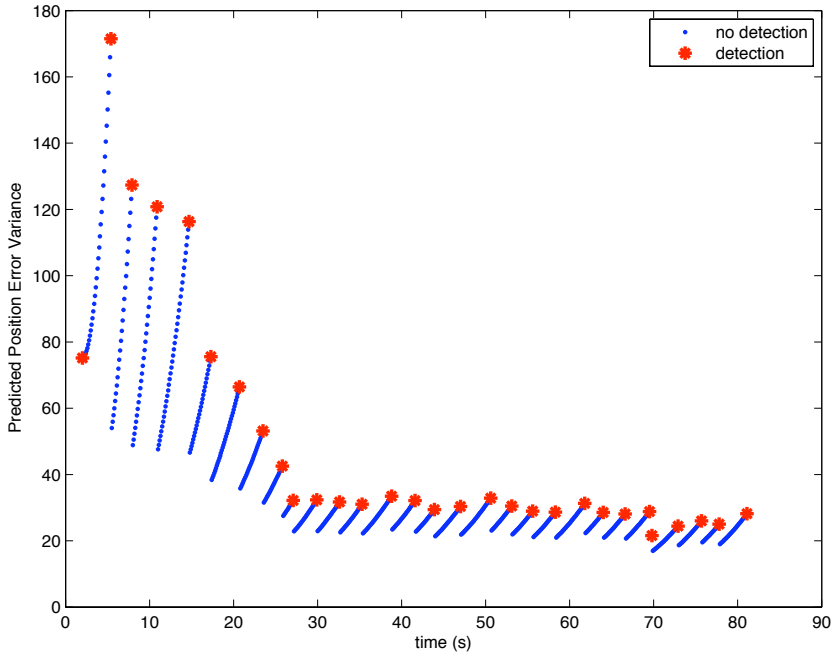


Figure 2.9: Case 1: Predicted variance $\mathbf{P}_n^-$ in the x (or equivalently, y) direction for target tracking with proximity sensors, $d = 15$, $q = 1$.

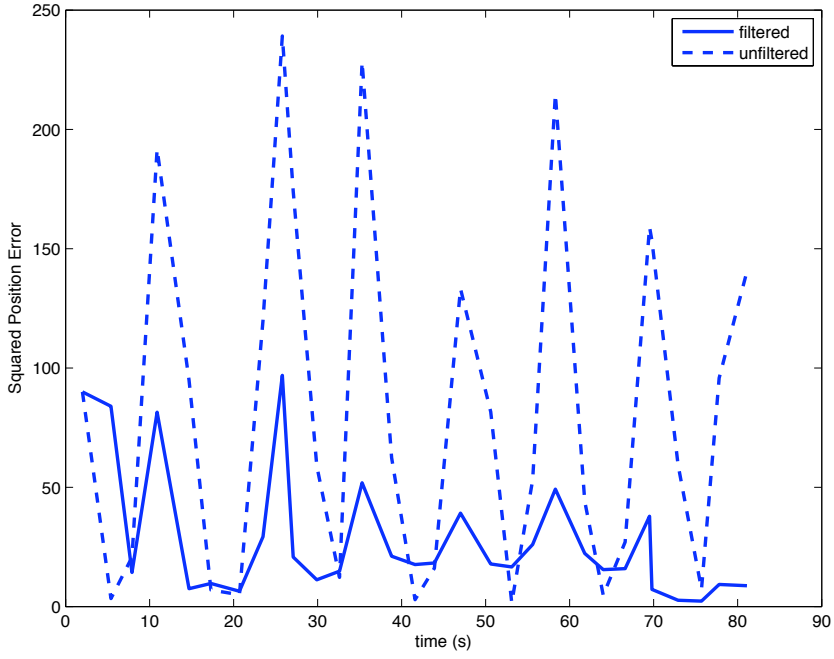


Figure 2.10: Case 1: Squared position error for target tracking with proximity sensors, $d = 15$, $q = 1$.

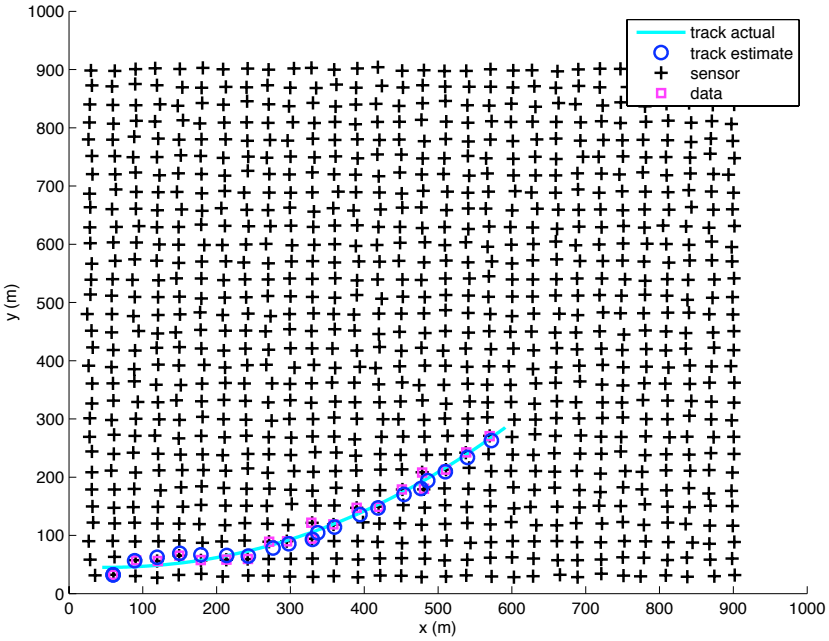


Figure 2.11: Case 2: Simulation of target tracking with proximity sensors, curved track, $d = 15$, $q = 1$.

design change that could be made to the tracking algorithm is to increase the assumed process noise. This change allows the Kalman filter to compensate for target motion that is more dynamic than assumed by the plant model.

2.7 Summary

The proximity sensor provides very little information about the target location, compared to sensors that detect the received power of a target or directional information. One advantage of using the proximity sensor is that it reduces the amount of data that must be shared among the nodes, increasing the overall power efficiency of the network. Also, as our simulations show, the Kalman filter is able to improve these position measurements to get better estimates of the target position. Another useful feature of the Kalman filter is the prediction error circles, which can be used in an energy-saving effort to select the set of sensor nodes to be active for tracking.

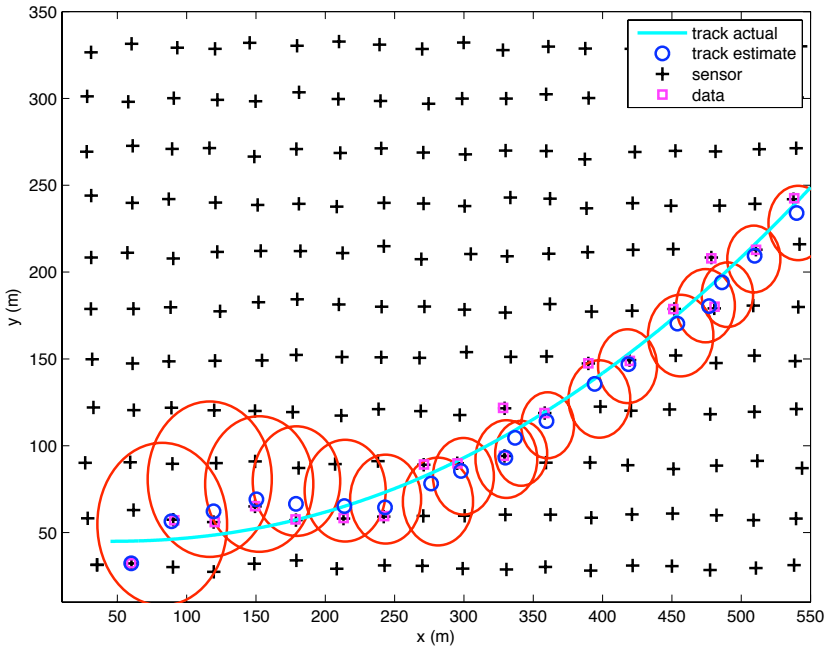


Figure 2.12: Case 2: Simulation of target tracking with proximity sensors, (zoomed version) with 3σ prediction circles, curved track, $d = 15$, $q = 1$.

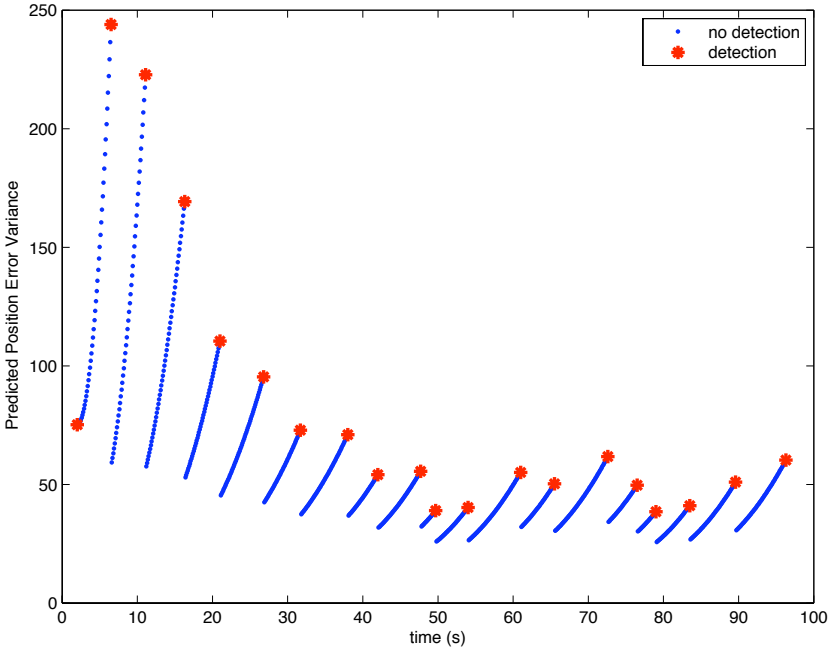


Figure 2.13: Case 2: Predicted variance $\mathbf{P}_n^-$ in the x (or equivalently, y) direction for target tracking with proximity sensors, curved track, $d = 15$, $q = 1$.

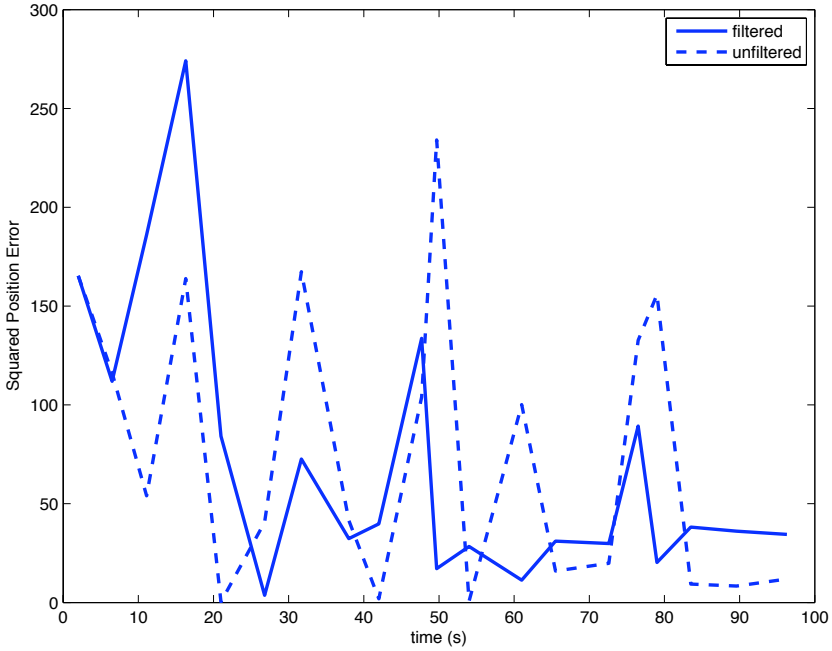


Figure 2.14: Case 2: Squared position error for target tracking with proximity sensors, curved track, $d = 15$, $q = 1$.

2.8 Acknowledgements

This chapter is, in part, a reprint of material published in the article: "Distributed Target Tracking in a Wireless Sensor Network," appearing in *Conference Record of the Fortieth Asilomar Conference on Signals, Systems & Computers*, ACSSC '06, IEEE 2006. The dissertation author was the primary researcher and author in the publication, and William Hodgkiss supervised the research.

Chapter 3

Sparse WSNs: Local Information-Based Power Management

In the previous chapter, we examined how a data-driven approach to energy efficiency in a multi-hop WSN affects a specific task of the sensor network. In this chapter, we consider the case of sparse WSNs, in which there are fewer nodes in the network, but transmission ranges are still short relative to the deployment area. We present a duty-cycling power management scheme for delay-tolerant networks that is designed to run on top of a MAC. First, we present the general idea of including local information in a sleep/wakeup decision. Then we choose a specific application to develop our sleep/wakeup strategy, and evaluate its effectiveness through simulation.

3.1 Introduction

In recent years, significant effort has been devoted to a class of mobile ad hoc networks (MANETs) known as delay- or disruption-tolerant networks (DTNs). DTNs are typically composed of nodes with intermittent connectivity, and mobility is often utilized to route data between nodes with no end-to-end connectivity. Applications such as vehicular ad hoc networks (VANETs), large-scale mobile sensor

networks, or networks in challenged environments can often suffer from low connectivity. Applying a DTN framework to such networks can enable communication to a greater extent.

Just as in traditional wireless sensor networks, energy conservation is a concern for delay-tolerant networks. Because data delivery in DTNs is characterized by long delays, it is important to extend the network lifetime. Mobile nodes are equipped with batteries of limited capacity, thus constraining the operation of the network. A power management mechanism designed specifically for DTNs would aid energy efficient communication.

Some power management mechanisms that have been proposed for DTNs use some global information about the network to determine when to put nodes in a low-powered sleep state. For example, Jun et al [JAZ05] used statistics about node inter-contact time and contact duration time to trade off contact discovery probability for energy. Others have used network traffic load information to determine a sleep schedule. However, in addition to the global network information, there is often a lot of local information available to aid in power management. For instance, nodes may have their own position/velocity information from GPS, packet queue length information, expiration time of contacts, history of contacts, or road traffic information (for VANETs). We propose using such information for nodes to decide when to be awake to transmit or receive, and when they can sleep.

For example, we consider a node that has no data to transmit, but may make itself available to relay data for another node. If a node's local information indicates that its probability of delivery is not good (e.g., it is far away from a known sink, it has not come into contact with many nodes), the node can decide to put the radio in a low-power sleep mode more often. With such a strategy, nodes actually utilize local information to reduce the amount of idle listening when they have low routing utility, thus conserving energy.

3.2 Related Work

Power management for MANETs and wireless sensor networks (WSNs) is a well-researched area. Although power management for MANETs and WSNs could be applied to a DTN framework, the focus is different because traditional MANETs assume end-to-end connectivity, whereas DTNs are characteristically sparse networks, in which nodes make infrequent contacts through mobility. There have been a few power management schemes proposed for DTNs, but they have not exploited the local information available to the extent that we have in our work. Also, the aim of our work is to minimize the energy consumption under the constraint of achieving a desired delivery ratio.

Jun et al. [JAZ05] proposed a synchronous, knowledge-based power management framework for delay-tolerant networks. They used global information to trade off contact discovery probability for sleep, such as inter-contact times and contact duration. The only local information used is knowledge of current contacts. Xi et al. [XCC07] proposed an asynchronous context-aware power management scheme (CAPM), which calculates the duty cycle based on the traffic load and node density to minimize energy. Our proposed scheme differs in that it is synchronous and our sleep schedule is built on a baseline duty cycle, which could be determined in a similar manner as CAPM.

The existing scheme that best takes advantage of local information is the adaptive exponential beacon period protocol (AEB) [CS09]. AEB is a synchronous protocol in which each node uses the knowledge of the most recent contact to lengthen the beacon period. This scheme is a precursor to our work, which seeks to exploit all useful local information to determine a sleep schedule. In our approach, we layer our power management scheme on top of AEB. We compare our performance to AEB only and show that our scheme outperforms AEB in energy for a given delivery ratio. Our approach also has more flexibility in tuning its performance to achieve a desired delivery ratio.

3.3 Local Node State for Delay-Tolerant Routing

Our proposal of using a node's local state information to do power management often goes hand in hand with the particular routing scheme. In this section, we present an overview of delay-tolerant routing and how it relates to local information.

In traditional MANET routing, network topologies are assumed sufficiently stable relative to the transmission speed, such that end-to-end routes can be established and used for data transmission. For DTNs, end-to-end routes are typically unavailable, so routing must be performed in an opportunistic manner. Specifically, at each node encounter, a node must make a decision about transmitting its data.

Most DTN routing falls under the category of multi-copy routing, in which a source node propagates multiple copies of a message through the network in an effort to reduce the delay and increase the overall delivery ratio. Usually, multi-copy strategies include some type of control to limit the storage and communication demands. Some of these strategies are *stateless*, in which no external information is used in routing decisions [VB00, SPR05], while others take into account information on the nodes' states, such as location or contact history data [LDS04, WW06]. A smaller portion of the research is devoted to single-copy routing, in which forwarding decisions can be made randomly or based on some utility function [SPR04]. If a single-copy routing scheme is well-designed and sufficient information is available to the nodes, the routing scheme can be competitive with multi-copy routing, while requiring significantly less network resources.

Whether multi-copy or single-copy, it is typically advantageous to use nodes' state information in a given DTN routing scheme to predict whether forwarding data to another node will improve data delivery. In the next section, we describe how we use that same local information for power management in such a way that routing performance is minimally affected.

3.4 Local Node State for Power Management

3.4.1 Network Model

We model our network as a collection of wireless nodes moving in a circular area. The node density is sufficiently low such that the network is generally disconnected. The nodes move according to the random direction (RD) mobility model [Bet01] with zero pause time, and nodes' velocity vectors reflect off the edge of the space. Our proposed power management is a clock-based scheme that assumes network-wide synchronization.

3.4.2 Baseline Beaconing Mechanism

We propose a framework for power management using a synchronous wakeup, in which nodes decide when to beacon and listen to make a connection with another node. As a baseline, we start with a modified AEB protocol [CS09], since it already utilizes some local information. This mechanism consists of a beacon window, a post-window space, and a wakeup period (Fig. 3.1), each of which is synchronized across all nodes. This network-wide synchronization is achieved independent of the beacon messages, which are described as follows. When a node wants to beacon, it waits a random time before sending its beacon message. A node sending a beacon-response then waits a random time before sending in the post-window space.

The lengths of the periods in the baseline cycle are set a priori, and they can be set to approximate a particular delivery performance, which is dependent on the given specifications of network dynamics (e.g., transmission range, node velocity, node density, motion model, etc.). Setting these period lengths is outside of the scope of our work. The actual beacon cycle doubles for each beacon opportunity without transmission, up to $2^{h_{max}}\times$ the original cycle. Our baseline only differs from the original AEB in that the communication is sender-initiated instead of listener-initiated. This results in slightly lower energy consumption, since only nodes with data send a beacon. Our power management is layered on top of this baseline beaconing.

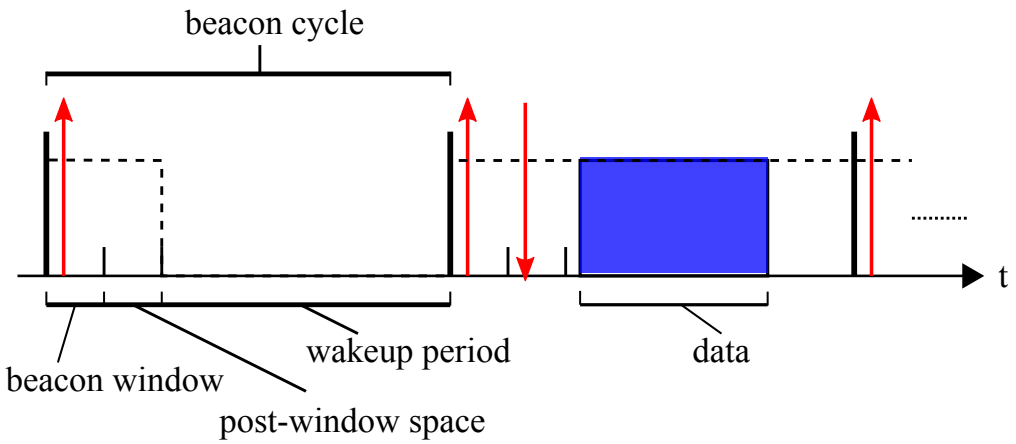


Figure 3.1: Baseline beaconing mechanism for DTN power management.

3.4.3 Local Information-Based Sleep

With the baseline AEB protocol in place, nodes can further decide at each beacon opportunity whether to wake up or continue to sleep. For our approach, we use the local information available to the node to make this decision.

A node makes an estimate of its ability to deliver data based on information available to it prior to any communication in the current beacon window. Examples of such information may be the packet queue length, packet expiration times, history of contacts, road traffic conditions, mobility map information, or battery level. These kinds of information are frequently proposed in the literature for use in delay-tolerant routing. Our scheme uses the same information to judge a node's ability to deliver data, and subsequently makes decisions for sleep and wakeup.

To describe our power management concept, we provide an example of how local information can be used to reduce energy consumption for a node with data, which we call a *data carrier*, and a node without data, which we call a *relay node*. For a data carrier, the local state can be more favorable or less favorable for successful delivery to the sink. An example of a favorable state is having a history of frequent contact with the destination. If a carrier is in a favorable state, it can reduce the frequency of beaconing and listening to detect other nodes, since it has a high likelihood of delivery and is in no need of a relay node. A carrier in an unfavorable state will beacon more frequently, to try to hand off to a node with a

more favorable state.

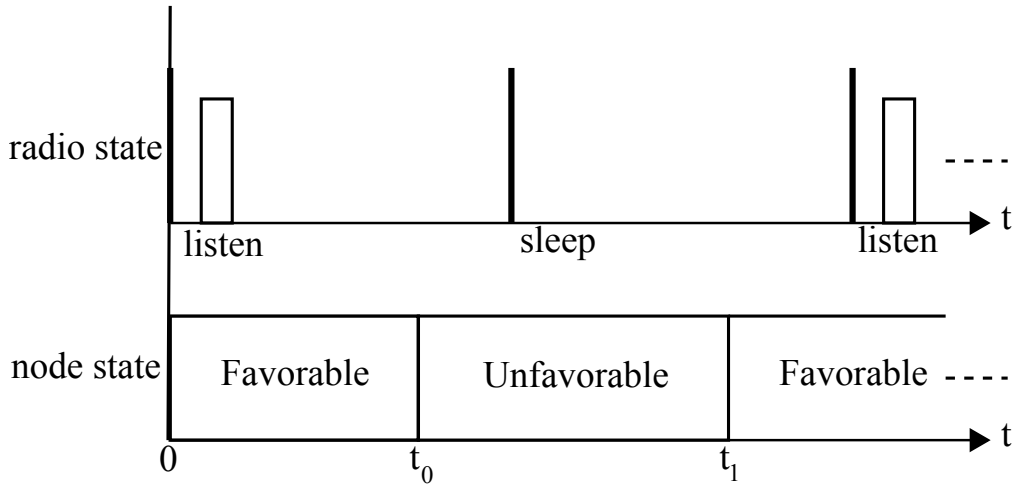
Relay nodes make themselves available to assist carriers in delivering data with a desired probability of success. A relay node with an unfavorable state has a lower likelihood of delivery and has little use in offering its relaying services, so it listens less frequently. When in a favorable state, a relay node will listen more frequently to take on data from carriers with a lower likelihood of delivery.

In Fig. 3.2, we illustrate the manner in which we run our local information-based power management mechanism on top of the baseline beaconing mechanism. For the purposes of this illustration, we assume a beacon at every cycle ($h_{max} = 0$). We first consider a relay node and illustrate its operation in Fig. 3.2a. At time 0, a relay node uses its local information to determine that it has a high enough probability of contacting the destination. Because it is in a favorable state, it will listen to assist an unfavorable node in need of a relay. In this case, it does not receive any data. At time t_0, the state of the node changed enough so that it is no longer considered favorable. It is unnecessary to make itself available since it has a low probability of delivery, so instead of listening, it chooses to sleep through the entire period. When the node becomes favorable again at time t_1, it listens at the next opportunity.

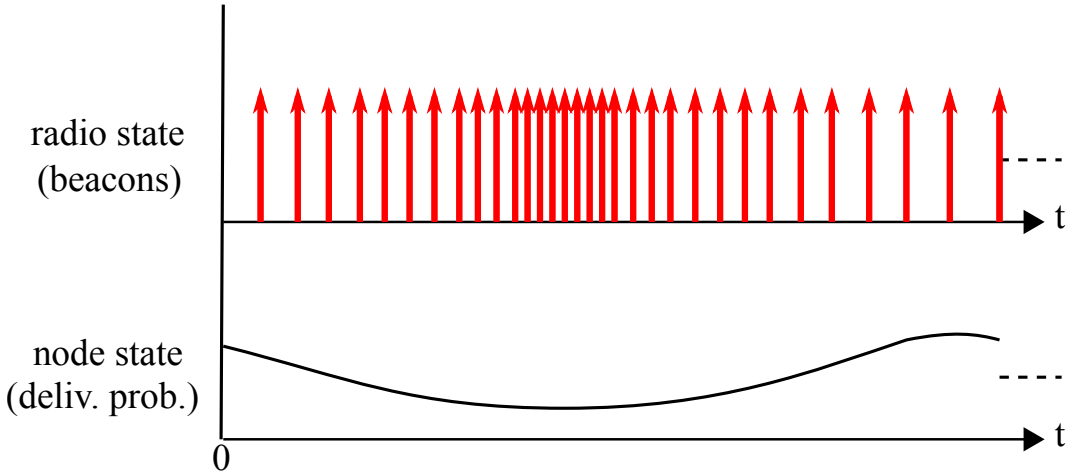
The strategy for data carriers is illustrated in Fig. 3.2b. Data carriers have to be concerned with achieving a desired delivery ratio for its data. For data carriers, the decision may not simply be to sleep when unfavorable and wake up when favorable. Rather, the task is to choose a suitable beaconing/listening rate, such that the data is delivered with a certain probability of success. A carrier with a less favorable state wakes up for more of the beacon windows because it wants to hand off to a more favorable node. A carrier with a more favorable state can sleep for more of the beacon windows because it is less likely to require a relay.

3.5 Geographic Convergecast Application

To illustrate our local information-based power management concept, we choose a single-copy, geographic routing scenario. In geographic routing, nodes



(a) Local information-based sleep for relay nodes.



(b) Local information-based sleep for carrier nodes.

Figure 3.2: Local information-based power management.

have knowledge of their own physical locations (e.g., via GPS, localization) and of the location of the destination, and they make routing decisions based on this information. This scenario frequently arises in systems such as mobile sensor networks or vehicular ad hoc networks (VANETs). This particular application is very rich in local information that is available at each node, so it is well suited for our power management scheme. We do not consider replication-based (epidemic) routing, since our power management is most efficient when nodes have empty queues. For simplicity, we consider a convergecast application, in which many source nodes send data to a single sink node.

In our geographic routing, the metric used to make handoff decisions is the rate of progress of the mobile node. When a node with a packet makes contact with another potential relay node, they compare their rates of progress toward the destination. This information is contained in the beacon and beacon-response packets. If the potential relay node has a higher rate of progress, the packet is handed to it, and it assumes the duty of delivering the packet. The equation for the rate of progress is

$$\dot{p} = v \cos\theta \tag{3.1}$$

where θ is the angle between the vector from node to destination and the velocity vector (Fig. 3.3). We assume the magnitude of the velocity vector v is constant. As a result, our routing metric performs identically to the motion vector (MoVe) algorithm for VANETs [LCGZ05], in which the angle θ is the routing metric.

For the geographic convergecast application, the local state metrics that we consider in our sleep strategies are *rate of progress* ($\dot{p}$), *proximity to sink* (d), and *packet expiration time* (t_{exp}). For these metrics, the favorable states include a higher rate of progress, closer proximity to the sink, and a longer packet expiration time. In the following section, we detail the methods for mapping these local states to a sleep/wakeup decision.

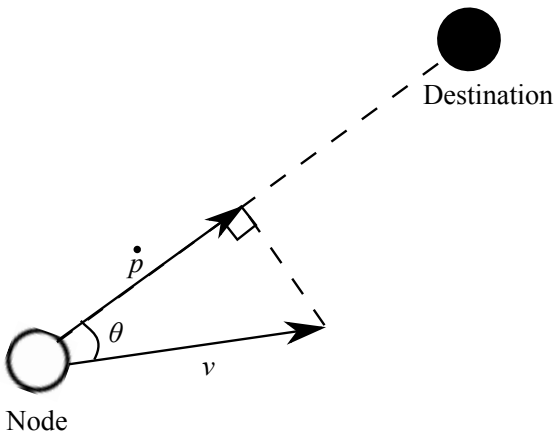


Figure 3.3: Rate of progress metric.

3.6 Power Management for Geographic Convergecast

The goal for our power management scheme is to minimize energy consumption while still achieving a desired packet delivery ratio. If a low enough target delivery ratio is chosen, a simple approach to minimizing the energy is to put all relay nodes to sleep and only wake up data carriers when in range of the sink node (since they have knowledge of their own and the sink's locations). For our purposes, we only consider target delivery ratios that are high enough to require some routing.

We express the node's local state as a combination of the three metrics, ($\dot{p}$,d,t_{exp}). We choose different approaches for relay nodes and data carriers when mapping the state to a sleep/wakeup decision. Relay nodes are not in possession of any packets, so t_{exp} is unknown to them, and they only know the state ($\dot{p}$,d). Even for this state space, it is still complex to evaluate a relay node's likelihood of data delivery. For simplicity, we only consider the state $\dot{p}$, and we define a threshold $\tilde{\dot{p}}$. Then the sleep/wakeup is decided as follows:

$$\text{relay node radio state} = \begin{cases} \text{Sleep} & , \text{ if } \dot{p} < \tilde{\dot{p}} \\ \text{Wake up} & , \text{ if } \dot{p} \geq \tilde{\dot{p}} \end{cases} \tag{3.2}$$

For data carrier nodes, our strategy is to set a beaconing/listening rate for

a given state ($\dot{p}$,d,t_{exp}). Similar to AEB, we choose beacon periods $T_{beac}(\dot{p}, d, t_{exp})$ that are powers of 2, so that carrier nodes have beacon periods that are multiples of one another. To choose an appropriate T_{beac}, a data carrier estimates its own delivery probability for various values of T_{beac}, and it chooses the lowest beacon rate that approximates the desired delivery probability. Since T_{beac} depends on t_{exp}, it must be computed at each time step, and the sleep/wakeup is decided as follows:

$$\text{carrier node radio state} = \begin{cases} \text{Sleep} & , \text{ if } t \pmod{T_{beac}} \neq 0 \\ \text{Wake up} & , \text{ if } t \pmod{T_{beac}} = 0 \end{cases} \tag{3.3}$$

where t is a global clock time step.

3.7 Pre-computing Delivery Ratio Using Markov Chain Model

Because we do not have closed-form expressions of the geographic delay-tolerant routing scenario, we cannot compute estimates of the delivery probability in real time. Instead, we compute the threshold $\tilde{\dot{p}}$ for relay nodes and the beacon periods $T_{beac}(\dot{p}, d, t_{exp})$ for carrier nodes prior to implementing the power management on the nodes.

To approximate the delivery ratio for the given design parameters $\tilde{\dot{p}}$ and $T_{beac}(\dot{p}, d, t_{exp})$, we model the geographic delay-tolerant routing system as a discrete Markov chain. We quantize the state values of $\dot{p}$ and d so that there are a finite number of states:

$$\dot{p} \in \{[-10, -9), [-9, -8), \ldots, [8, 9), [9, 10]\} \tag{3.4}$$

$$d \in \{[0, 100), [100, 200), \ldots, [1300, 1400), [1400, 1500]\} \tag{3.5}$$

Our state vector, $\mathbf{x}_t$, represents the distribution of data packets throughout the system. With the above quantization, there are 300 possible combinations

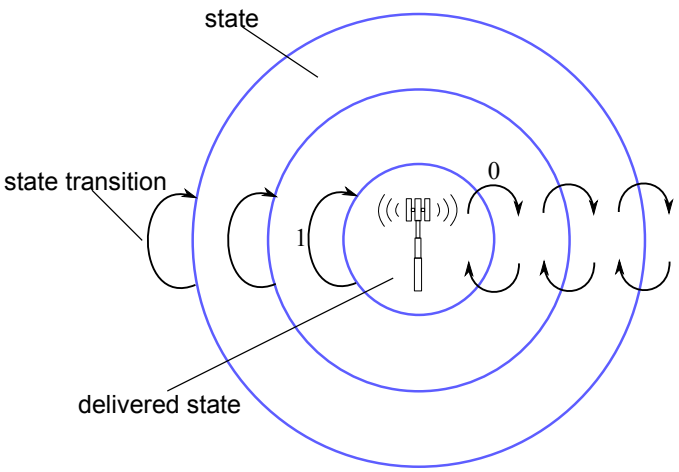


Figure 3.4: Markov chain model for geographic convergecast.

of $(\dot{p}, d)$ pairs, so $\mathbf{x}_t$ is a 300-element row vector. To model the progress-based routing and the network dynamics, we have two 300×300 transition matrices, the routing matrix $\mathbf{\Phi}_{rt}$ and the motion matrix $\mathbf{\Phi}_{mv}$. If a node is awake for a packet handoff opportunity, we update the system using $\mathbf{x}_t = \mathbf{x}_t \mathbf{\Phi}_{rt}$. At each time step, the dynamics of the system makes the update $\mathbf{x}_{t+1} = \mathbf{x}_t \mathbf{\Phi}_{mv}$.

To model the data arrival at the sink, we set some of the elements $\phi_{(\dot{p},d)_m,(\dot{p},d)_n}$ in the transition matrices $\mathbf{\Phi}_{rt}$ and $\mathbf{\Phi}_{mv}$ so that any packet that enters a state with a d that is in transmission range of the sink does not leave that state (Fig. 3.4). Formally, we define those matrix elements as

$$\phi_{(\dot{p},d)_m,(\dot{p},d)_n} = \begin{cases} 1 & , \quad \text{if } m = n \\ 0 & , \quad otherwise \end{cases} \tag{3.6}$$

for the $(\dot{p}, d)_m$ in which $d < r$, where r is the transmission range. To get s_t, the delivery ratio at a given time t, we sum up the states in $\mathbf{x}_t$ that have d in range of the sink:

$$s_t = \sum_{d<r} x_{(\dot{p},d),t} \tag{3.7}$$

3.7.1 Computing Transition Matrices

To model the progress-based routing in the transition matrix $\mathbf{\Phi}_{rt}$, we first made the simplifying assumption that handoffs occur in the same relative proximity d. Then we modeled the handoffs as the transition $(\dot{p}_m, d) \rightarrow (\dot{p}_n, d)$, where $\dot{p}_m < \dot{p}_n$. This is computed as:

$$\begin{aligned} \phi_{(\dot{p},d)_m,(\dot{p},d)_n} &= p_{mc} P(\text{random node's progress} = \dot{p}_n) + \\ & \quad p_{mr} P(\text{random node's progress} = \dot{p}_n) \mathbf{1}(\dot{p}_n > \tilde{\dot{p}}) \end{aligned} \tag{3.8}$$

where p_{mc} and p_{mr} are the probability of meeting a carrier or relay during one time step, respectively, and $\mathbf{1}(\dot{p}_n > \tilde{\dot{p}})$ is the indicator function for when the receiving state's $\dot{p}$ is greater than the threshold $\tilde{\dot{p}}$. p_{mc} and p_{mr} are computed analytically using methods from Spyropoulos et al[SPR06]. The expressions are as follows:

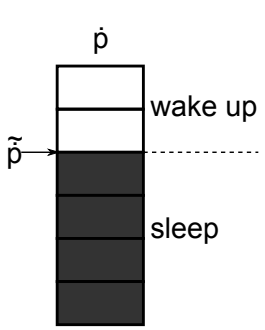
$$p_{mc} = 1 - (1 - \frac{2rv\hat{v}_{rd}}{A})^{p_c N} \tag{3.9}$$

$$p_{mr} = 1 - (1 - \frac{2rv\hat{v}_{rd}}{A})^{p_r N} \tag{3.10}$$

where $\hat{v}_{rd}$ is the normalized relative velocity for the RD mobility model, p_c is the probability that a random node is a data carrier, and $p_r = 1 - p_c$ is the probability that a random node is a relay node. The values of p_c and p_r are dependent on the network settings. Because data converges to fewer nodes over time, the p_c and p_r terms are actually time-dependent values, but we model them as constant. For our study, these values were obtained by simulation. The expressions for p_{mc} and p_{mr} are approximate, in that they do not account for initial conditions and edge effects.

The probability of remaining in the state $(\dot{p}, d)_m$ is

$$\phi_{(\dot{p},d)_m,(\dot{p},d)_m} = 1 - \sum_{(\dot{p},d)_n \neq (\dot{p},d)_m} \phi_{(\dot{p},d)_m,(\dot{p},d)_n} \tag{3.11}$$



(a) Setting thresholds.

ṗ → / d ↓						
	4	4	8	8	16	16
	4	4	8	8	16	16
	2	2	4	4	8	8
	2	2	4	4	8	8
	1	1	2	2	4	4
	1	1	2	2	4	4

(b) Setting beacon periods for t.

Figure 3.5: Setting threshold and beacon periods.

Table 3.1: DTN Simulation Parameters.

Parameter	**Value**
Packet Lifetime	2500 s
Field Radius	1500 m
Transmission Range	200 m
Node Velocity	10 m/s
RD Time Epoch	10 s

The network dynamics transition matrix (Φ_{mv}) was generated by Monte Carlo simulation of a single node. This captures the crossover from one geographic zone to the next, as well as the random change in direction as determined by the RD mobility model. The simulation parameters are shown in Table 3.1.

3.7.2 Computing the Progress Threshold $\tilde{\dot{p}}$

To initiate the Markov chain, we choose $\mathbf{x}_0$ that assumes a uniform distribution of packets over the circular area and a uniform distribution over the starting progress. To calculate the routing transition matrix, we consider the possible $\tilde{\dot{p}}$ taken from the set $\{-10, -9, \ldots, 9, 10\}$ and calculate $\mathbf{\Phi}_{rt}$ for each. We then step through the Markov chain with both $\mathbf{\Phi}_{rt}$ and $\mathbf{\Phi}_{mv}$ for all 2500 seconds of the packet lifetime, and compute the delivery ratio $s_{\tilde{\dot{p}},2500}$ (Eqn. 3.7). We choose the $\tilde{\dot{p}}$ that minimizes the normalized energy consumed, which we estimate as

$$e = p_{car} + (1 - p_{car})P(\dot{p} > \tilde{\dot{p}}) \tag{3.12}$$

where p_{car} is the probability that a node is a data carrier, which we obtain through simulation. e is the proportion of energy consumed by data carriers (always wake up) plus the proportion of energy of consumed by relays (wake up if above $\tilde{\dot{p}}$) (Fig. 3.5a).

3.7.3 Computing the Beacon Period T_{beac}

After choosing the minimum energy threshold $\tilde{\dot{p}}$, we calculate the beacon period for each of the 300 possible Markov states for a carrier. For this calculation, we further quantize time into members of the set $\{[0, 100), [100, 200), \ldots, [2400, 2500)\}$, such that there are 7500 possible carrier states, ($\dot{p}$,d,t_{exp}).

We consider two approaches to pre-computing the T_{beac}: one using a *local* delivery estimate and the other using a *global* delivery estimate. The first approach is to choose the longest beacon period that achieves the desired delivery ratio, based on the current local state ($\dot{p}$,d,t_{exp}). If the delivery ratio is not achievable, the beacon period is the same as the baseline beaconing mechanism. The initial condition $\mathbf{x}_0$ is simply 1 for the current Markov model state $x_{(\dot{p},d)}$ and 0 for the remaining states. We use the $\mathbf{\Phi}_{rt}$ corresponding to the minimum energy threshold $\tilde{\dot{p}}$, and compute the delivery probability for a beacon period $T_{beac}(\dot{p}, d, t_{exp}) = 1, 2, 4, \ldots$, until the delivery probability is no longer satisfied (Fig. 3.5b). Note that we only update the state with $\mathbf{\Phi}_{rt}$ every $T_{beac}(\dot{p}, d, t_{exp})$ time steps, in accordance with the beaconing (Expr. 3.13).

$$\mathbf{x}\mathbf{\Phi}_{rt}\mathbf{\Phi}_{dyn}^{T_{beac}}\mathbf{\Phi}_{rt}\ldots \tag{3.13}$$

With the local estimate approach, the overall delivery ratio is likely to overshoot the desired delivery ratio by a wide margin. This occurs because the T_{beac} is chosen such that the calculated delivery ratio will be greater than the desired delivery ratio for each local state individually, so the average over all states will also be greater. To get a closer approximation of the desired delivery ratio, we consider a second approach.

For the second approach, we first compute the largest beacon period that achieves the desired delivery ratio for each state with an expiration time in $[0, 100)$, using the first approach. Then we compute the beacon period for states with an expiration time in $[100, 200)$, but account for the beacon periods for the expiration time in $[0, 100)$. Because there are different beacon periods for different states, this requires a routing transition matrix $\mathbf{\Phi}_{rt}(t)$ that is time dependent. Depending on the time step and each state's beacon period, some rows assume beaconing and some rows do not.

For example, assume state ($\dot{p}_1$,d_1,$[0, 100)$) has a beacon period of 1 and state ($\dot{p}_2$,d_2,$[0, 100)$) has a beacon period of 2. The row of $\mathbf{\Phi}_{rt}(t)$ that corresponds to state 1 will compute the state transitions due to routing for every time step. However, for state 2, the row of $\mathbf{\Phi}_{rt}(t)$ will only compute the state transitions due to routing for every 2 time steps. For the other time steps, the row is set to that of the corresponding row in the identity matrix (the diagonal element is set to 1, the rest are zeros). $\mathbf{\Phi}_{rt}(t)$ is defined for every t, and the state is operated on by the routing transition matrix at each time step.

After beacon periods are computed for states with expiration time in [100, 200), we repeat the procedure for states with an expiration time in $[200, 300)$, and keep repeating while incrementing the expiration time up to $[2400, 2500)$. Using this global approach, we can reduce the margin by which we overshoot the desired delivery ratio, when comparing with the local estimate approach.

3.8 Simulation Evaluation

We implemented our delay-tolerant routing and power management scheme in the network simulator ns-2. We used the simulation parameters from Table 3.1, and for the baseline beaconing, we used AEB with $h_{max} = 3$. At time 0, there are 40 data carriers and 0 relay nodes (carriers become relays when they unload their data), and the sink is located at the center of the circular area. In each case, the simulation time was 2500 seconds.

Table 3.2: Delivery Ratio for LI-PM+AEB, Geographic Convergecast for DTNs, Threshold Only.

$\tilde{\hat{p}}$	**Calculated** s	**Simulated** s
None	98.01%	96.77%
2 m/s	97.86%	97.55%
4 m/s	97.58%	97.80%
6 m/s	97.00%	97.86%
8 m/s	95.49%	97.29%
10 m/s	76.55%	74.50%

3.8.1 Progress Threshold Only

For our initial set of simulations, we investigated the effect of implementing a threshold on the progress metric for all relay nodes. In this case, $T_{beac} = 1$ for all nodes. First we computed the delivery ratio using the Markov model of the DTN for different threshold levels. Then we simulated the scenario using ns-2, averaging over 1000 simulations. We see in Table 3.2 that the resulting calculated and simulated delivery ratios after 2500 seconds have similar values. Sources of error in the calculated scenario have been described in the previous section.

We have plotted the simulated delivery ratio against delay in Fig. 3.6. We observe that for most delay values, higher thresholds lead to lower delivery ratios since nodes sleep more and have fewer routing opportunities. However, by the end of the simulation, the delivery ratios converge, and sometimes a higher threshold can have a slightly higher delivery ratio. For example, in Table 3.2, a threshold of 4 m/s has a slightly higher delivery ratio (97.80%) than that of a 2 m/s threshold (97.55%). A possible cause for this phenomenon is that routing is more likely to transmit to a node farther away from the sink (Fig. 3.7), since we only consider rate of progress as the routing metric.

We have also plotted the simulated delivery ratio against energy consumed in Fig. 3.8. We observe that relative to no threshold, energy consumption can be about 4 times less when we use a threshold for a desired delivery ratio of about 97%. Also, as we choose higher thresholds, the energy consumed decreases, since

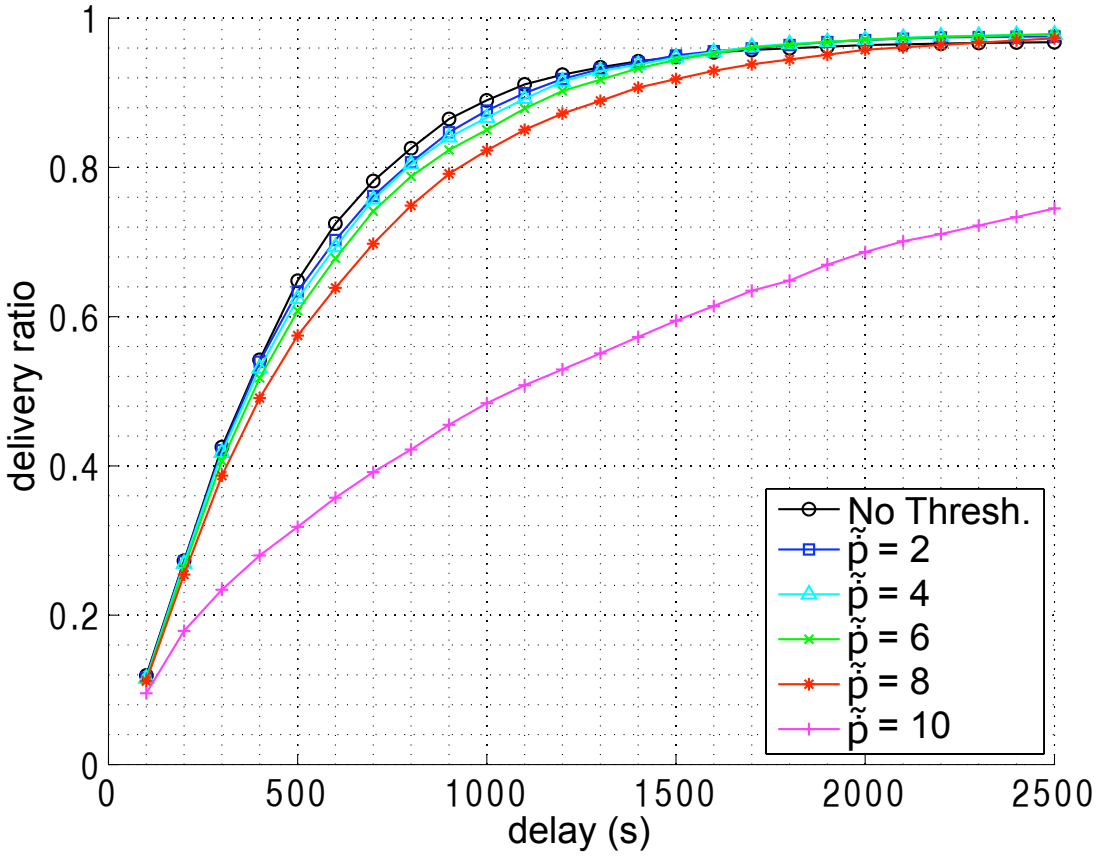


Figure 3.6: Delivery Ratio vs. Delay for LI-PM+AEB, Geographic Convergecast for DTNs, Threshold Only.

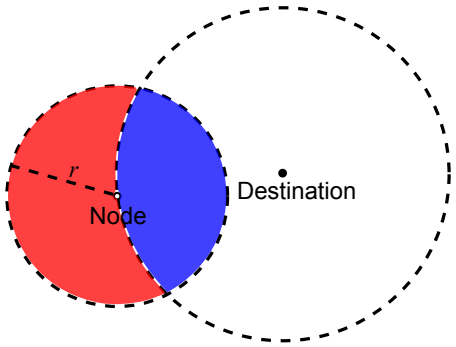


Figure 3.7: Relay nodes are more likely to be farther away (red area) from the destination.

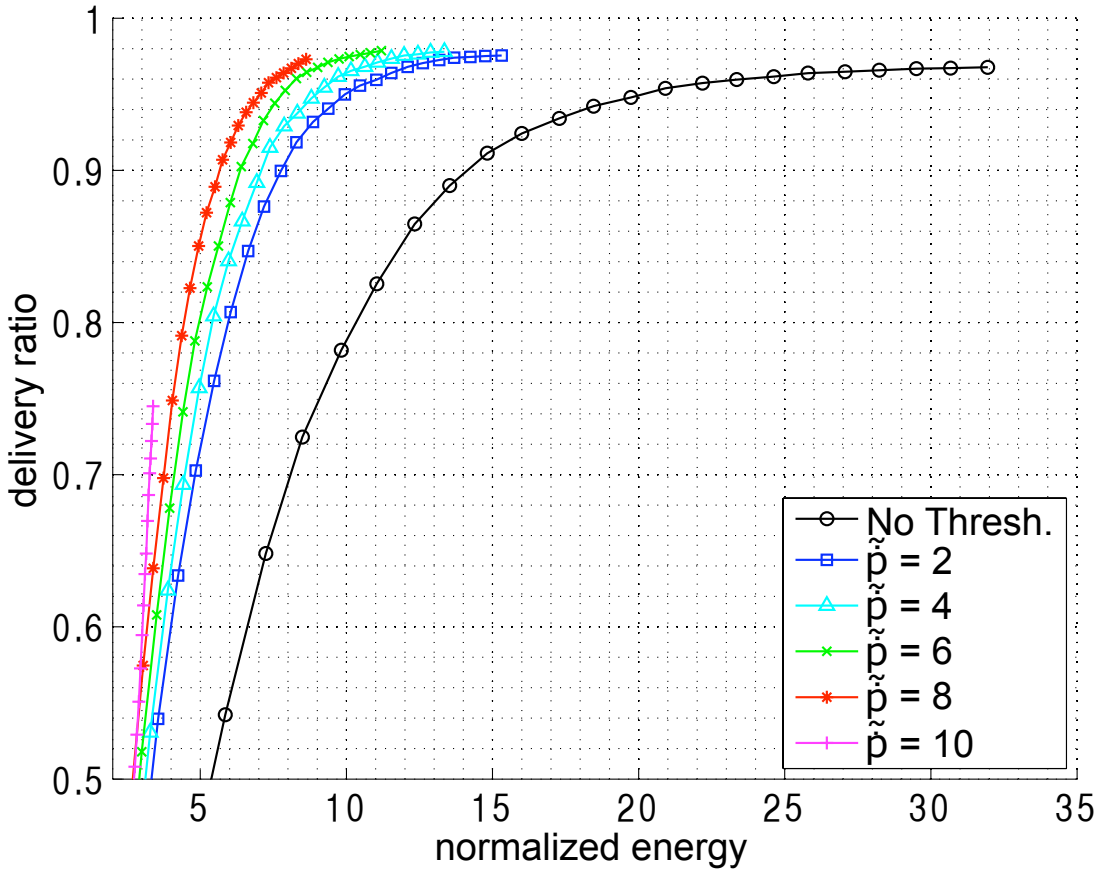


Figure 3.8: Delivery Ratio vs. Energy Consumption for LI-PM+AEB, Geographic Convergecast for DTNs, Threshold Only.

nodes are more likely to be under the threshold, and thus sleeping more often. Because we ran our simulation long enough, most of the delivery ratios are similar for large changes in threshold. Only with a threshold at the maximum progress of 10 m/s is the delivery ratio drastically different, in which case, relay nodes always sleep. Therefore, if the desired delivery ratio is less than 74.20%, relay nodes can always sleep.

3.8.2 Threshold and Beacon Period, Local Estimate

For the second set of simulations, we set both the threshold and beacon period to achieve the delivery estimate locally. We calculated the $\tilde{p}$ and $T_{beac}(\dot{p}, d, t_{exp})$ for target delivery ratios $s = 0.95, 0.85, 0.75$, and provide the results in Table 3.3. We also show the calculated s using the Markov model, which can have up to 6% higher value than the target value. This is partly due to choosing the beaconing period such that each of the 300 states individually achieve the target s, so that the collective s is much higher. Also, the granularity of beaconing rates is not fine enough to approximate the target s. However, the beacon periods were chosen to be exponential for the synchronization properties with other nodes. The simulated s are also provided in the table. As in the previous subsection, they are in the ballpark of the calculated s, but some approximations are made in the Markov model that result in these differences.

In Fig. 3.9, we have plotted the delivery ratio vs. delivery delay for AEB Only and for our local information-based scheme running on top of AEB (LI-PM+AEB). These results are averaged over 500 simulations. We observe that the delivery ratios for the LI-PM+AEB schemes increase more slowly as time progresses relative to AEB Only Schemes. This trend is due to the t_{exp} being favorable when it is high, so LI-PM sleeps more early in the simulation, thus trading off latency for energy.

In Figs. 3.10-3.11, we have plotted the delivery ratio vs. energy consumption for the same set of schemes. The normalized energy is simply the awake time per node. When we compare LI-PM+AEB, 95% with AEB Only, $h_{max} = 3$, the delivery ratio is similar, but the energy consumed is on the order of 4 times less.

Table 3.3: Power Management Settings for LI-PM+AEB and AEB Only, Geographic Convergecast for DTNs, Local Estimate.

Target s	$\tilde{\dot{p}}$	**Average** T_{beac}	**Calculated** s	**Simulated** s
95%	8 m/s	1.01 s	95.49%	97.64%
85%	9 m/s	1.34 s	92.99%	97.12%
75%	10 m/s	2.05 s	76.45%	74.20%

Table 3.4: Delivery Ratio for AEB Only, Geographic Convergecast for DTNs.

h_{max}	**Simulated** s
3	96.77%
4	95.16%
5	91.27%
6	66.35%

With such rich local information available to the nodes, we are able to design an intelligent sleep/wakeup schedule that still achieves a high delivery ratio. Also, our sleep/wakeup allows for greater flexibility for achieving a desired delivery ratio. AEB Only does not have the same flexibility, as seen by the large gap in delivery ratios as h_{max} is incremented (Table 3.4). However, we do not observe great accuracy for the desired delivery ratio for this local estimate approach. In addition, the target delivery ratio of 75% is not achieved because the calculated delivery ratio for a threshold of 10 m/s was over 75%, but the simulated delivery ratio was actually under 75%.

3.8.3 Threshold and Beacon Period, Global Estimate

For our third simulation, we again set both the threshold and beacon period to achieve the delivery ratio, but this time we use the global estimate. Specifically, we estimate the delivery ratio for each local state using the beacon periods for states with lower t_{exp}. The beacon period is chosen such that the overall delivery ratio, starting from the current state and using all the beacon periods for states

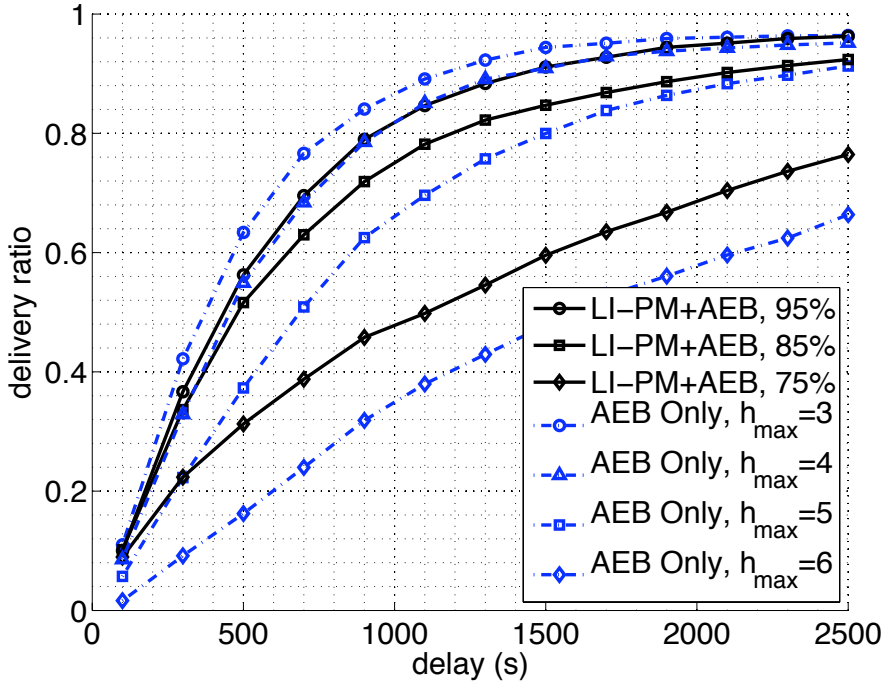


Figure 3.9: Delivery Ratio vs. Delay for LI-PM+AEB and AEB Only, Geographic Convergecast, Local Estimate.

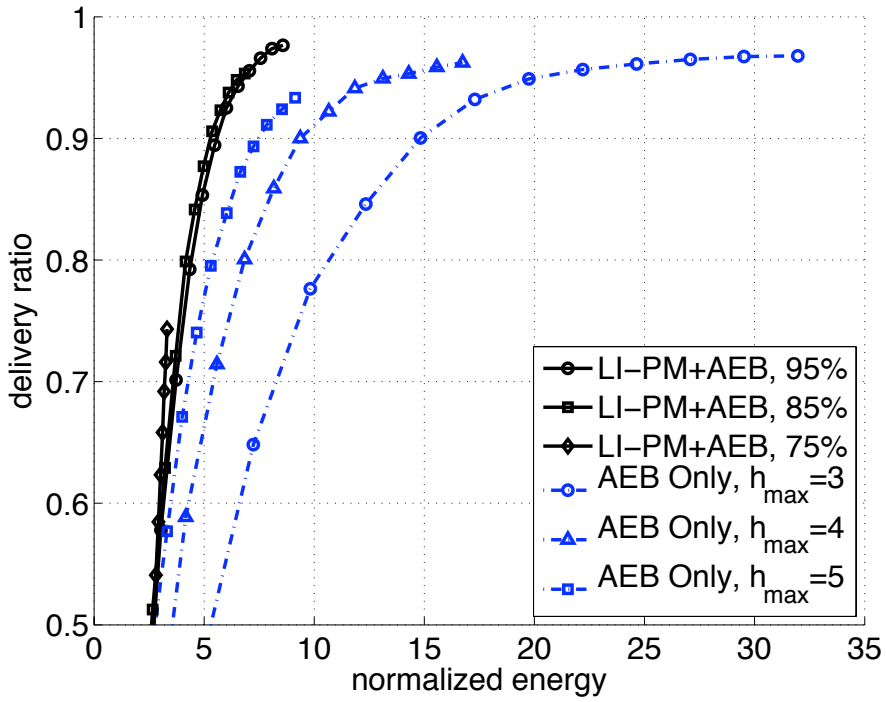


Figure 3.10: Delivery Ratio vs. Energy Consumption for LI-PM+AEB and AEB Only, Geographic Convergecast, Local Estimate.

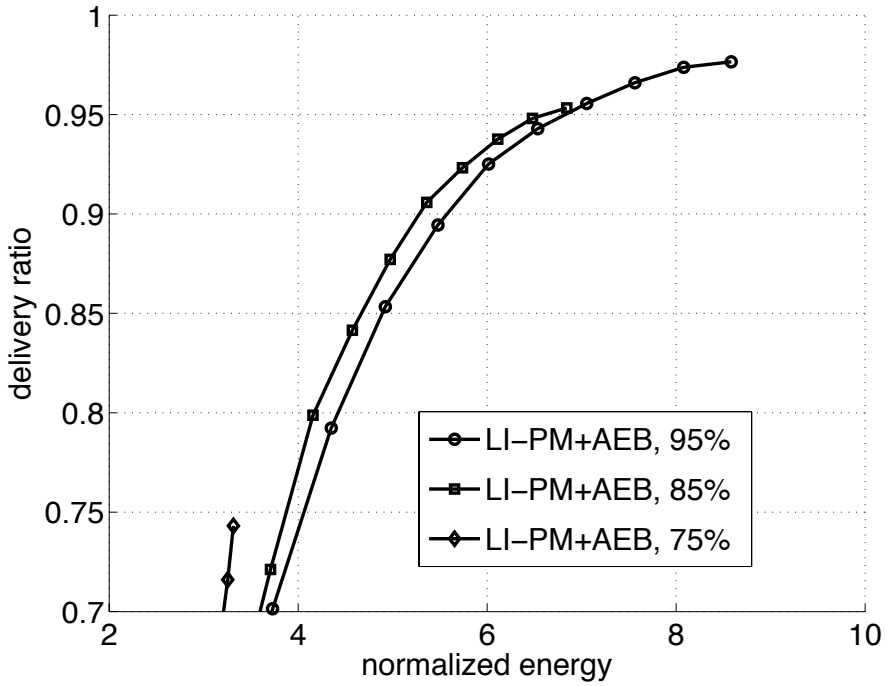


Figure 3.11: Delivery Ratio vs. Energy Consumption for LI-PM+AEB (zoomed), Geographic Convergecast, Local Estimate.

Table 3.5: Power Management Settings for LI-PM+AEB, Geographic Convergecast, Global Estimate.

Target s	$\tilde{\dot{p}}$	**Average** T_{beac}	**Calculated** s	**Simulated** s
95%	8 m/s	2.76 s	94.95%	97.36%
85%	9 m/s	9.74 s	89.46%	94.26%
75%	10 m/s	3.45 s	76.26%	73.76%

with lower t_{exp}, achieves the target delivery ratio. We show our calculated results in Table 3.5.

In Fig. 3.12, we have plotted the delivery ratio vs. delivery delay for LI-PM+AEB using the local and global estimates. These results are averaged over 500 simulations. We observe that the global estimate reduces the delivery ratio further, since we are not just calculating an estimate of the delivery ratio at each local state. Again, we have delayed the delivery further, with latency being traded off for energy.

In Fig. 3.13, we have plotted the delivery ratio vs. energy consumption for the same set of schemes, and we have also plotted the threshold only curves to show the improvement that LI-PM+AEB yields. Here we see that both global and local approaches save energy over the threshold only approach, and the global estimate does not overshoot the target delivery ratio as much as the local estimate. As expected, the global approach consumes less energy than the local approach. For 95% and 85%, the target delivery ratio is achieved. Again, for 75%, the target delivery ratio is not achieved because the calculated delivery ratio for a threshold of 10 m/s was under 75%.

3.9 Summary

In this work, we have demonstrated the concept that local information can be used in designing a power management scheme for delay-tolerant networks. We have shown a procedure for determining a sleep/wakeup strategy for a geographic convergecast application using local information. We propose a threshold-based

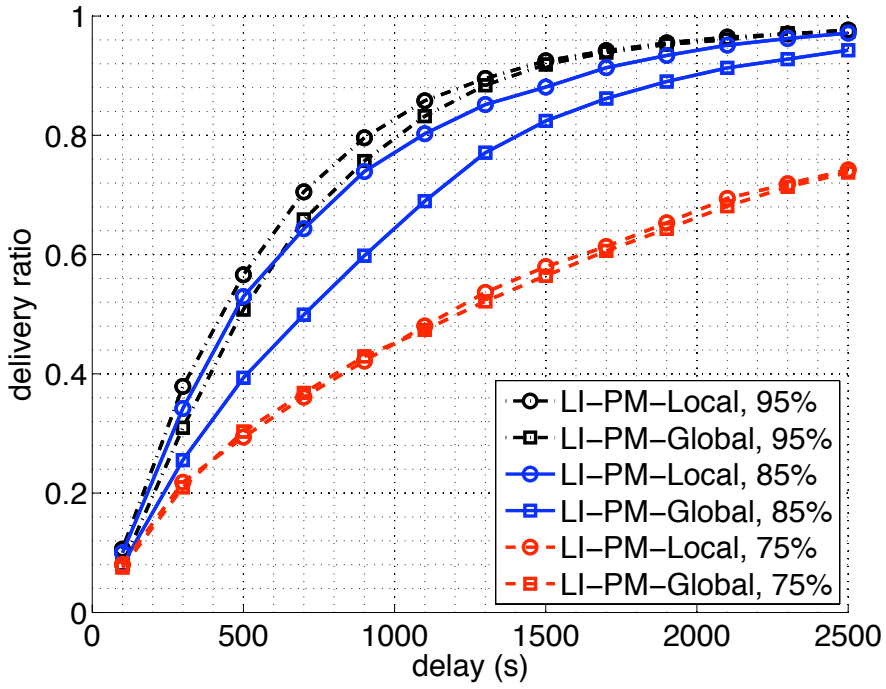


Figure 3.12: Delivery Ratio vs. Delay for LI-PM+AEB, Geographic Convergecast, Global vs. Local Estimate.

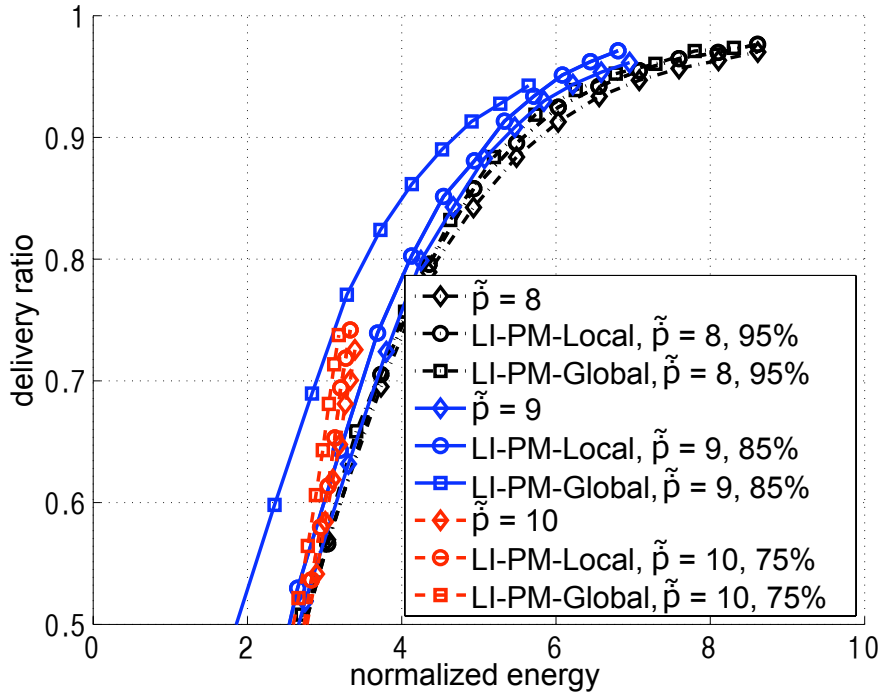


Figure 3.13: Delivery Ratio vs. Energy Consumption for Threshold Only, LI-PM+AEB, Local, and LI-PM+AEB, Global, Geographic Convergecast.

strategy for relay nodes and a beacon period strategy for data carrier nodes. Our simulation results show that for this application, our scheme layered over an AEB scheme approximates a desired delivery ratio while reducing energy consumption by 4× over AEB only. Our heuristic approach calculates a rate of progress threshold for relay nodes, and sets beaconing rates for data carriers.

3.10 Acknowledgements

This chapter is, in part, a reprint of material accepted for publication in *IEEE Workshop on Mobile Computing and Emerging Communication Networks (MCECN'10)* under the title "Local Information-Based Power Management in a Delay Tolerant Network." The dissertation author was the primary researcher and author in the publication, and Curt Schurgers supervised the research.

Chapter 4

Dense, Small-Scale WSNs: A Hybrid MAC/Routing Solution for Convergecast

In the previous chapter, we looked at a duty-cycling approach to sparse WSNs, specifically a local information-based power management scheme for delay-tolerant networks. In this chapter, we consider bringing the relatively few nodes in close proximity, yielding a dense, small-scale WSN. Again, we apply a duty-cycling approach that is appropriate for this particular type of network. To develop our approach, we first present a study on the the typical topologies of small-scale networks. Then we use our insights from the study to develop a combined MAC/routing approach. We evaluate our approach using simulation to compare with an existing solution. We also provide experimental evaluation and show the effect of a real-world wireless channel.

4.1 Introduction

The use of networked autonomous vehicles is of great value to many sensing applications such as environmental monitoring or disaster response. For such applications, the ability to dispatch these vehicles to a remote (and often hostile) location is critical. Since these vehicle applications are of a different nature than

traditional sensor networks, a different approach should be taken to the networking and communication of such systems. Before we detail our approach, we first draw out the design objectives by setting the problem in the context of a specific sensing application.

In this example, a small number of unmanned aerial vehicles (∼10 UAVs) are equipped with sensors (e.g., smoke, humidity) to detect some substance in the atmosphere, and the system's task is to report the sensor data to a base station for further processing. Because we are dealing with planes in flight, they will be communicating over relatively long distances, on the order of a kilometer. Therefore, the typical low-power, short-range radio technologies used for sensor networks (e.g., 802.15.4) are inappropriate for this application. Instead, this system requires a higher-powered radio technology, such as 802.11. We have conducted tests with a pair of UAVs with 802.11 radios, demonstrating the ability to communicate over ranges of up to a mile. In addition, these radios are powered by batteries with significant weight and volume, which contribute to the planes' payload. The allowable payload is a critical constraint, limiting the battery that can be mounted on the plane. Given the limited on-board batteries and high-powered radios, it is important to design our networking strategy with energy efficiency as our primary constraint.

Although we have described a specific example here, we believe this scenario covers the more generic set of vehicle-based sensing applications. To define the vehicle-based networking problem, we identified certain characteristics of such systems, which, in some cases, set them apart from the well-studied traditional sensor networks. One characteristic of these networks is that they typically have a *convergecast* traffic pattern, in which many nodes report data to a single sink node. Another characteristic is that these networks have a *dynamic topology*, since the vehicles can be highly mobile. The impact on data routing is that routes expire quickly and the opportunity for route reuse is limited. Lastly, these networks are of a *small scale*, in which the number of nodes is on the order of ten to a few tens of nodes. The reason for such small network sizes is that the cost to manufacture and deploy these vehicles is much higher than the cost for small static sensor nodes,

thus limiting the scale of such networks. Therefore, scalability is not a requirement for the networking strategy, as is often assumed for traditional sensor networks. Instead, the need for scalability can be traded off for a more efficient solution. Our networking problem is to provide energy-efficient communication for small-scale, convergecast mobile networks. In our solution, we leverage the fact that these networks are small-scale for more efficient operation.

The first step in our design approach was to gain some insight into the degree of routing that is actually required for small-scale networks. We conducted a preliminary topology study, in which we simulated a network of 15 nodes in a 100 meter radius disk sending to a sink node located at the center. For each node, a transmission range was fixed such that it has connectivity with other nodes in that range. Then, with the network's node connectivity given, we measured the minimum distance in hops of each node from the sink. We gathered these statistics over many realizations of random node deployments, and we calculated the percentage of times that nodes are n hops from the sink for various transmission ranges (Fig. 4.1). As a systems designer, we would like to use the information from Fig. 4.1 to make the best design choice for a radio with a specified transmission range. We observe that for transmission ranges less than 80 meters, some nodes do not have a route to the sink, which is inadequate for real-time reporting. Therefore, to avoid having a disconnected network, a range of at least 80 meters is required. However, if we choose an 80 meter range, a majority of the nodes are already just 1 hop from the sink. Furthermore, we know that, in reality, radio transceivers do not have precise transmission ranges, since the actual range is dependent on the environment. To compensate for such uncertainties, the system would have to be overdesigned such that a longer-range radio is chosen. In selecting a longer range, we observe from Fig. 4.1 that 100% of nodes very quickly come into 1-hop range of the sink. So in practice, nodes will mostly be in direct communication with the sink. Therefore, full-scale routing is not necessary. Note that this result only comes about due to the small number of nodes in the network.

Although we have observed that in most cases nodes are one hop away from the sink, there are still some rare situations for which a multi-hop route is

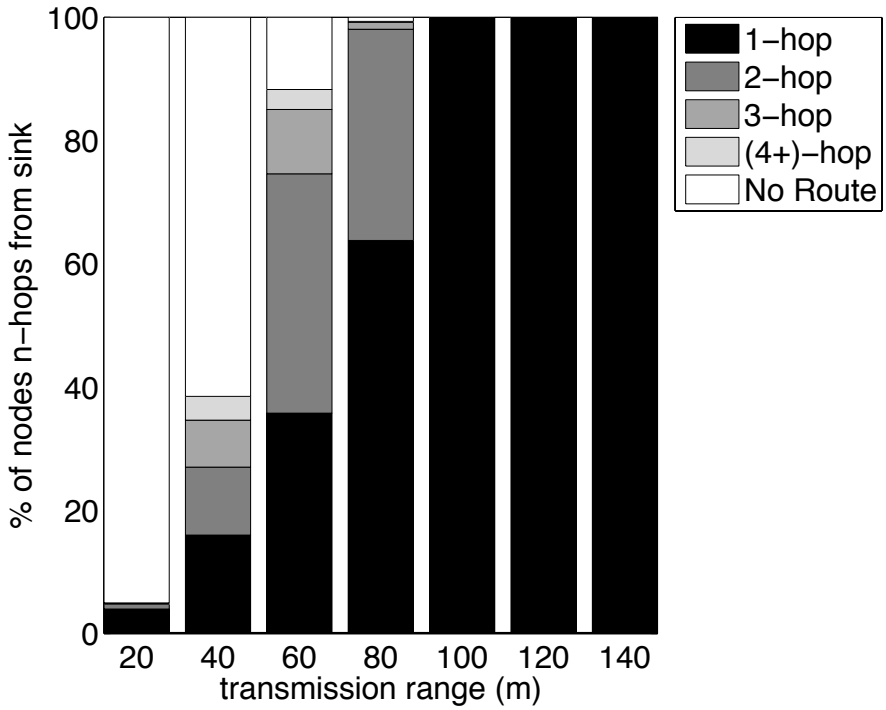


Figure 4.1: Connectivity vs. transmission range, 15 node network placed in a disk of 100 meter radius.

required. It is important to design our solution to also be able to handle such scenarios. Taking our UAV application as an example, a group of UAVs may be sensing a large plume of smoke located in some expected area, in which all nodes have a 1-hop route to the base station. However, it is possible that there is a smaller peripheral cloud of smoke that one of the UAV detects farther away from the main plume, and direct communication with the base station is not possible. It is still important to report such information, so a multi-hop route would be needed.

Our observation is that for these small, vehicle-based sensor networks, nodes are mostly in 1-hop range of the sink, but in rare cases, a node may need a multi-hop route to report data to the sink. Our networking approach exploits this knowledge of the typical route lengths so that the communication can be optimized for energy efficiency. To do this, we eliminate the need for a separate routing layer, and instead incorporate the routing functionality directly into the medium access control (MAC). As a result, we have a networking solution that behaves mainly as an efficient MAC protocol, but it retains the ability to multi-hop route to account for such rare situations.

4.2 ConverSS: Hybrid MAC/Routing Solution

ConverSS is a combined MAC and routing solution for *reliable* and *energy-efficient convergecast* for *small-scale*, *mobile* networks. ConverSS is designed specifically for cases where most nodes are one hop from the sink, and where mobility limits the opportunity for reusing routes. For clarity purposes, we first present a brief overview of the protocol operation and provide two illustrative examples, before describing the full features of the protocol. The pseudocode of the full protocol is included at the end of the chapter.

For many of these real-time sensing applications, sensor-generated data must be sent to a sink periodically. We focus on one cycle of our protocol operation, or a *sending interval*, in which each node has one data packet to deliver. A sending interval occurs periodically with the *data arrival period* (Fig. 4.2).

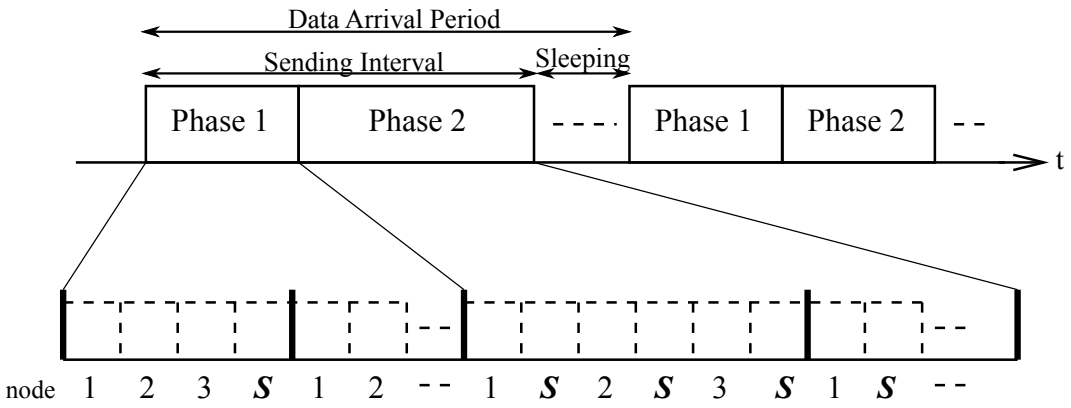


Figure 4.2: ConverSS two-phase approach.

Because these are small networks, it is feasible to use a fixed-assignment TDMA MAC, in which each node is assigned a dedicated time slot for sending. This MAC requires that the number of nodes in the network be fixed at the system initialization. However, this setup is sufficient for most sensing missions, which have a small, consistent team of vehicles. Time synchronization for the MAC is done using GPS, which has been used in existing systems and shown to achieve accuracy within 30ns [JOW+02, ZC98].

4.2.1 Two-Phase Approach of ConverSS

The idea behind ConverSS is that it functions primarily as a fixed-assignment MAC, but includes a fallback mechanism to accommodate multi-hop routing. We capture this idea in a two-phase approach (Fig. 4.2), which efficiently handles the most common cases in the first phase, and then if necessary, deals with all other cases in the second phase. In Fig. 4.2, we label the owner of each sending slot below every time slot, with the sink's slot indicated by an ***S***.

Phase 1 operates under the assumption that nodes are one hop from the sink, so each node attempts to send directly to the sink in its sending slot. Since there is no routing in this phase, nodes do not need to listen in the other nodes' slots. They can instead be placed in sleep mode, in which nodes turn off the radio and thus consume very low power. The sink follows by broadcasting a Delivery Status Message (DSM), which contains information on which nodes' packets have

been delivered to the sink. All nodes listen to this slot to discover if their data has been delivered. In case of packet errors, more frames are allotted in which nodes can retransmit any undelivered data. By the end of Phase 1, packets from all 1-hop nodes should have been received by the sink. If this includes all nodes, as we expect in most cases, then they all go to sleep until the next sending interval. The operation described is efficient because there is no route setup, and nodes only send in their own slots and do not need to listen in the other slots. This will turn out to be the main difference with layered approaches, since a route must be established and the MAC will always assume that routing may be needed, resulting in idle listening.

It is possible that there are still undelivered packets after Phase 1 because of packet errors or nodes being out of range of the sink. If that is the case, a second phase is necessary to do a best-effort data delivery. In Phase 2, any nodes whose packets were not delivered perform a type of controlled flooding, in which nodes broadcast their data and receiving nodes can rebroadcast to try to deliver it to the sink. The mechanisms used to control the flooding and limit redundant transmissions will be illustrated in examples and in Section 4.3.2.

The reason for using flooding rather than a route setup followed by data transmission relates to the presence of asymmetric links. Studies have demonstrated the problem of asymmetry in radio propagation, in which a link is stronger in one direction than in the reverse [ZHKS06, SAZ07]. In typical routing, only symmetric links can be used because a handshake is required before a packet can be sent over a link. Flooding, however, does not require a handshake, thus enabling the use of asymmetric links in routing the data to the sink. Since these are small networks, routes with asymmetric links may be the only ones available. Therefore, with more options for routing, data delivery has an improved chance of success.

After the two phases have completed, the network can go to sleep until the next sending interval, when the next packet is ready for delivery. Given the small number of nodes, the two-phase sending interval is typically short enough such that the network topology will be stable during that time. Because no routes are assumed prior to Phase 1, the protocol is robust to changes in topology in

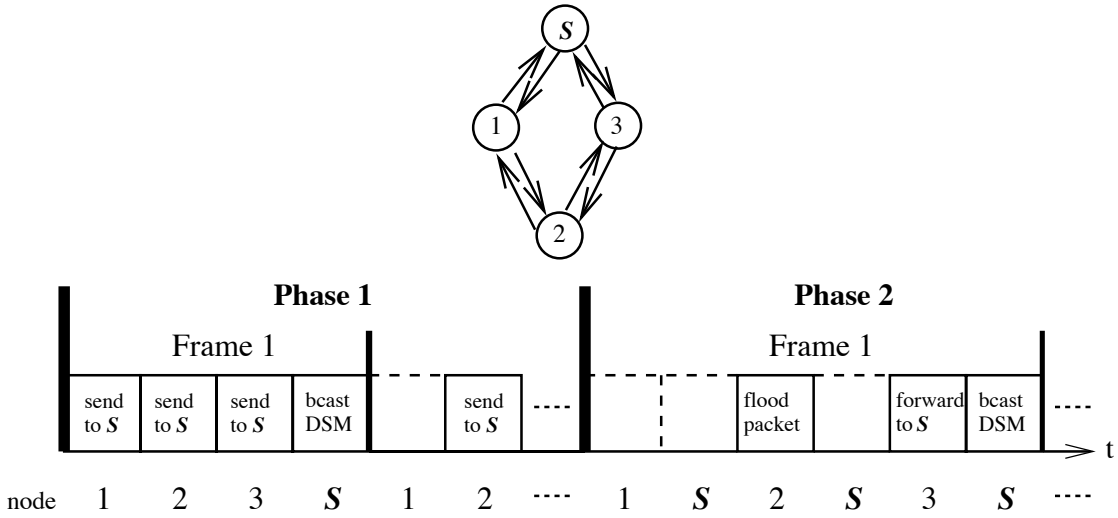


Figure 4.3: ConverSS Example 1–Simple Network.

between sending intervals. Only in rare cases will changes to the topology during the sending interval affect the delivery rate, and the system can recover from these by the next sending interval. The cost of such rare occurrences is small relative to the gains in energy efficiency that the protocol design affords.

Example 1–Simple Network

We first provide an example using the simple network topology shown in Fig. 4.3, in which all but one node have a 1-hop route to the sink, and the remaining node is two hops from the sink. In Phase 1, the nodes transmit unicast packets to the sink in turn, and conserve energy by going to sleep in each others' slots. The sink receives the data from all but node 2, and broadcasts this information in a Delivery Status Message. Nodes that receive the DSM, i.e., nodes 1 and 3, learn whose data have been received. Since node 2 did not receive the DSM, it attempts to retransmit in the remaining Phase 1 frames, but with no success.

In Phase 2, nodes whose packets have not been delivered (e.g., node 2) flood their data. Up until this point, nodes 1 and 3 have not received from node 2 since they were not listening in that slot in Phase 1. By receiving the DSM, nodes 1 and 3 know that node 2 does not have a route to the sink, so they listen in its slot. Nodes 1 and 3 receive the flooded packet, and node 3 forwards the message

to the sink. Note that in Phase 2, the frame structure changes and each sending slot is followed by the sink's slot, so that a DSM can be immediately sent. In this example, the sink sends the DSM after node 3, node 1 then learns that node 2's packet has been delivered, and node 1 can simply remove that packet from its queue. The DSM indicates that all nodes have delivered their data successfully and can go to sleep. Although not shown in the figure, node 3 sends a DSM to node 2, telling it to go to sleep.

Example 2–More Complex Network.

Now we look at a more complex example to illustrate some of the other features of ConverSS. The network shown in Fig. 4.4 is the same as in Example 1 but adds a node with a unidirectional link to the sink (node 5), and a node that is three hops away from the sink (node 4). In Phase 1, the nodes transmit unicast packets to the sink in turn, and the sink receives from nodes 1, 3, and 5. The sink then broadcasts the DSM, which is received by nodes 1 and 3. Nodes 2, 4, and 5 do not receive a DSM, so they retransmit for the remainder of Phase 1.

In Phase 2, nodes that believe their packets have not been delivered (e.g., nodes 2, 4, and 5) must flood their data and listen in all other slots. Nodes that received the DSM (e.g., nodes 1 and 3) now listen to nodes whose packets have not been delivered, namely nodes 2 and 4. Node 2's flooded data is received by nodes 1, 3, and 4. As in Example 1, node 3 forwards it to the sink, the sink broadcasts a DSM, and node 1 knows to remove node 2's packet from its queue. While node 3 forwards this data, node 2 overhears it and assigns node 3 as its parent node for future reference. The function of this will become apparent shortly.

Node 4 floods its packet, which is received by node 2. Instead of flooding, node 2 now knows to forward the packet directly to its parent. This parent assignment is a feature of ConverSS that is used to limit the amount of flooding. Node 4 follows suit by overhearing node 2 forward in the next frame and assigning node 2 as its parent. If there happened to be another node sending to node 4, node 4 would simply forward the packet to its parent.

Node 5's flooded data is received by the sink, which has already received

the data in Phase 1 and therefore drops the packet. Node 5 cannot receive a DSM because of the unidirectional link. It rebroadcasts a number of times to account for packet errors, and then it removes its own packet from the queue.

Once node 4's data has been delivered, the sink sends a DSM indicating that all packets have been delivered and nodes can go to sleep. Although not shown in the figure, node 3 sends a DSM to node 2 telling it to go to sleep, and node 2 sends a DSM to node 4. Node 5 does not receive a DSM and is idle for a period of time. All nodes have an idle timer running such that if it expires, the node goes to sleep.

4.3 ConverSS Protocol Features

In this section, we elaborate on the features illustrated in the previous examples, and describe the details of the protocol operation at work in the two-phase structure of ConverSS.

4.3.1 ConverSS Header

ConverSS employs a special header structure shown in Fig. 4.5. It consists of the following header fields: Packet Type, Packet Sequence Number, Source Address, Destination Address, Delivery Status Bitmap, and Acknowledgment Bitmap. The Delivery Status Bitmap indicates the source node's knowledge of which nodes' data have been successfully delivered. The Acknowledgment Bitmap indicates from which nodes it has received since its last sent packet. Every data packet uses this header. A Delivery Status Message is simply the ConverSS header with no payload.

4.3.2 Controlled Flooding

To reduce the amount of redundant transmissions during Phase 2, we implement the following two features that control the flooding of data.

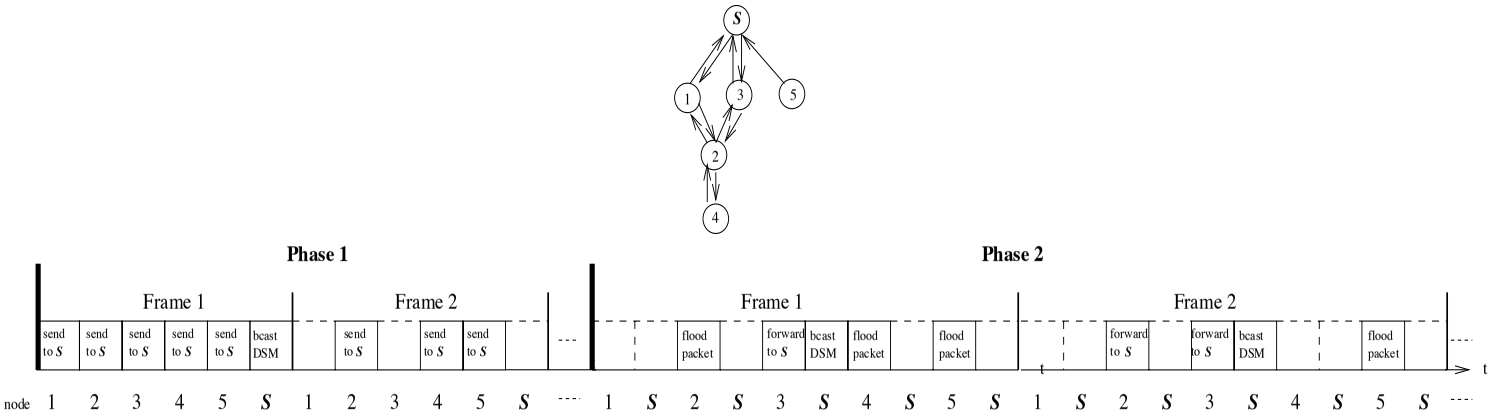


Figure 4.4: ConverSS Example 2–More Complex Network.

Type	SeqNo	SourceAddr	DestAddr	Delivery Status Bitmap	ACK Bitmap

Figure 4.5: ConverSS packet header.

Immediate Delivery Confirmation

Phase 2 utilizes a different frame structure in which a sending slot for the sink is inserted after each node's sending slot (Fig. 4.2). This structure allows the sink to immediately follow transmissions with a DSM, confirming delivery of the packet and stopping any other nodes in range of the sink from trying to deliver the same flooded packet. Since we expect nodes to mostly be in range of the sink for these vehicle applications, this frame structure should be very effective in limiting flooding and conserving energy.

Implicit Routing for Flooding Nodes

Although Phase 2 is the fallback to flooding, not all nodes perform flooding. Instead, an implicit routing allows some nodes to forward data upstream in the direction of the base station. We call any node with a route to the sink a *non-flooding node*.

At the start of Phase 2, nodes that had successful deliveries in Phase 1 do not flood because they know they are in the one-hop neighborhood of the sink. Instead, they forward any received data directly to the sink. Nodes outside of the sink's one-hop neighborhood start Phase 2 as flooding nodes. If a flooding node can establish a bidirectional link with a non-flooding node, then the flooding node has also realized a route. From that point on, the node can unicast forward any packets to the non-flooding node, thus becoming a non-flooding node.

To establish a bidirectional link with a non-flooding node, a flooding node tries to overhear a unicast transmission, which would confirm the incoming link. To confirm that the outgoing link exists, the non-flooding node must indicate that it has received from the flooding node. It does so in the acknowledgment bitmap (ACKB) in the header of the overheard transmission (Fig. 4.5). Then the flooding node can assign the other node as its *parent*, to which it forwards any packets in the

future. This completes the transformation of a flooding node into a non-flooding node. We note that not all nodes will become non-flooding, but they make this transformation when possible.

4.3.3 Phase 2 Listening Schedules

To reduce the amount of idle energy consumed in Phase 2, nodes construct listening schedules to determine in which slots to turn on the radio and listen. To construct these schedules, a Delivery Status Bitmap (DSB) field is included in the ConverSS packet header. The DSB distributes knowledge of which nodes' packets have been delivered to the sink, with a "0" indicating "not delivered" and a "1" indicating "delivered" (Fig. 4.5). Every node maintains a local DSB with the most current knowledge of the network.

At the start of Phase 1, every node sets its local DSB to be all 0's, and they only listen to the sink's slot. When the sink receives a packet, the sink sets the corresponding node's bit to 1. The sink sends a DSM containing the new DSB, and receiving nodes update their local DSB by performing a bitwise OR operation with the DSB field. This DSB is embedded in the header of every packet sent, so that overhearing nodes can update their local DSBs.

At the start of Phase 2, nodes listen to the slots of nodes that have a 0 in the DSB, in case they have to forward the data. If a node receives a data packet, it adds the source node as a *child*. If a flooding node overhears a non-flooding node acknowledge it in the ACKB, it assigns the node as its *parent*. Each node can only associate with one parent during each sending interval. The overall listening rule for each node is to listen to its parent and children, as well as nodes that correspond to a 0 bit in the DSB. As packets are received at the sink in Phase 2, the sink continues to set the originating node's bit to 1 and send out a DSM.

4.3.4 Sleeping Mechanism

We implement a sleeping mechanism so that when all packets have been delivered, nodes can sleep until the next sending interval. A node knows that all

packets have been received by the sink when its local DSB is all 1's. If the node has children, it broadcasts a DSM to update their DSB fields, and then the node goes to sleep. If a node does not have a parent and cannot otherwise get any updates to the DSB, an idle countdown timer is used. This timer operates such that a node that has not sent or received for a time T_{CD} goes to sleep until the next sending interval, since it assumes that it is not needed for forwarding.

4.4 Simulation Evaluation

To evaluate the performance of ConverSS, we implemented the protocol in the network simulator ns-2. For comparison purposes, we have also implemented two other protocols. To provide some insight into the best possible performance, we use an *Ideal* protocol, which uses a fixed-assignment TDMA MAC and an omniscient routing layer. In this protocol, nodes have knowledge of routes omnisciently, and can thus coordinate sending and listening schedules without any actual communication. In terms of performance, the energy consumption is minimum, and the latency is minimum for a fixed-assignment TDMA.

The second protocol that we compare to is a traditional *Layered* protocol. For routing, it utilizes a simplified version of one-phase pull directed diffusion [SHGE04]. Directed diffusion is a data-centric scalable routing framework that is commonly studied in sensor networks. We implemented a "slim" version of the protocol, which expects only one type of data.

The sink initiates the *Layered* protocol by flooding the network with an *interest*, or a data query. The interest expires at the end of each sending interval. Upon receiving the interest, nodes set up gradients according to which neighbor it received from first, and packets are sent along those gradients toward the sink. When the sink has received all data, it floods the network with a message telling nodes to go to sleep. For the MAC, this protocol uses a fixed-assignent TDMA. Each node has a sending slot, and nodes listen in all other slots. Because we use a time-slotted MAC, nodes that are not sending or receiving do not need to be idle for an entire time slot. Instead, nodes can listen at the start of the slot, and if no

Table 4.1: Power consumption of the ORiNOCO IEEE 802.11b PC Gold Card (11 Mbps).

Mode	**Transmit**	**Receive**	**Idle**
Power Consumption	1400 mW	950 mW	805 mW

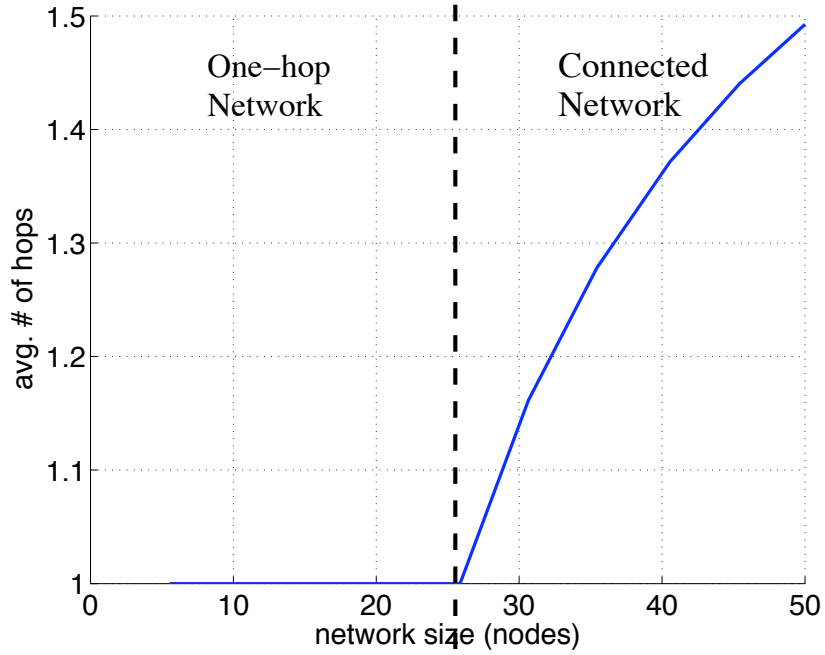
header is received, nodes can sleep for the remainder of the slot. We handle idle listening in the same manner for our *ConverSS* protocol.

For these simulations, one packet containing 64 bytes of data is generated at each node for every sending interval. To evaluate energy consumption, we used the power consumption specifications from the ORiNOCO IEEE 802.11b PC Gold Card, shown in Table 4.1 [SBS02]. We choose these specifications because 802.11 is the radio technology used in our UAV application.

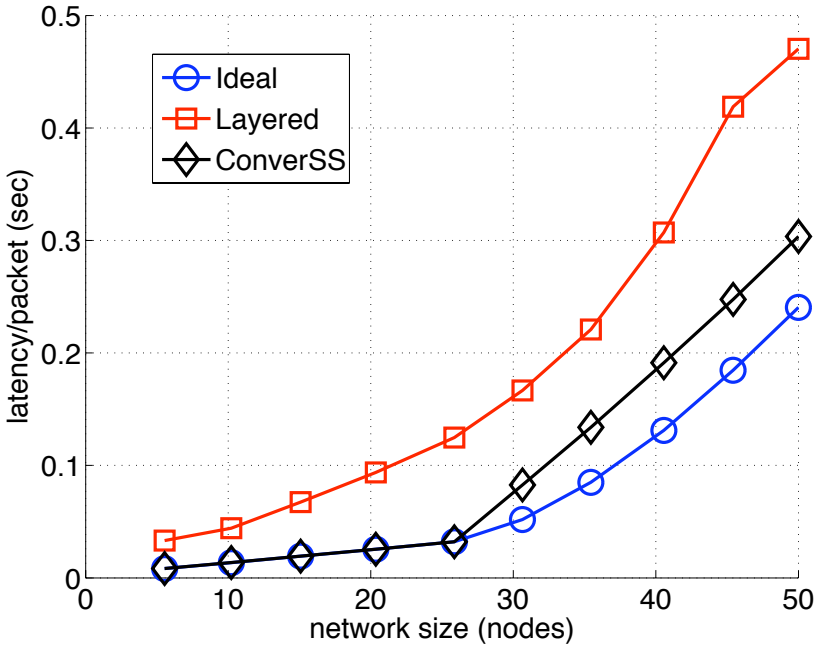
4.4.1 Varying Network Size

In the first simulation, we compared the different protocols for various network sizes. Each node is given an error-free transmission range of 100 meters. (We will investigate the effects of non-uniform range later.) Nodes are uniformly randomly deployed in a disk at a fixed node density of 0.0008 $\frac{\text{nodes}}{\text{m}^2}$. We vary the area of the disk centered around the sink, resulting in networks of different scales. Only nodes with a route to the sink are included in the simulation. Fig. 4.6 shows the average number of hops per node, the average latency, and the energy consumed per packet.

We note that for network sizes up to 25 nodes, all nodes are in one-hop range of the sink (Fig. 4.6a). We first comment on the results for these one-hop networks, for which *ConverSS* is optimized. For *ConverSS* and *Ideal*, the latency is small since the data is delivered in the first frame, and the latency for *Layered* is a bit more because the gradient to the sink must first be established (Fig. 4.6b). The energy consumption for *ConverSS* is very close to *Ideal* since all nodes immediately send to the sink and do not need Phase 2. *ConverSS* only consumes 10% of the energy of *Layered* (Fig. 4.6c), since layered protocols pay the

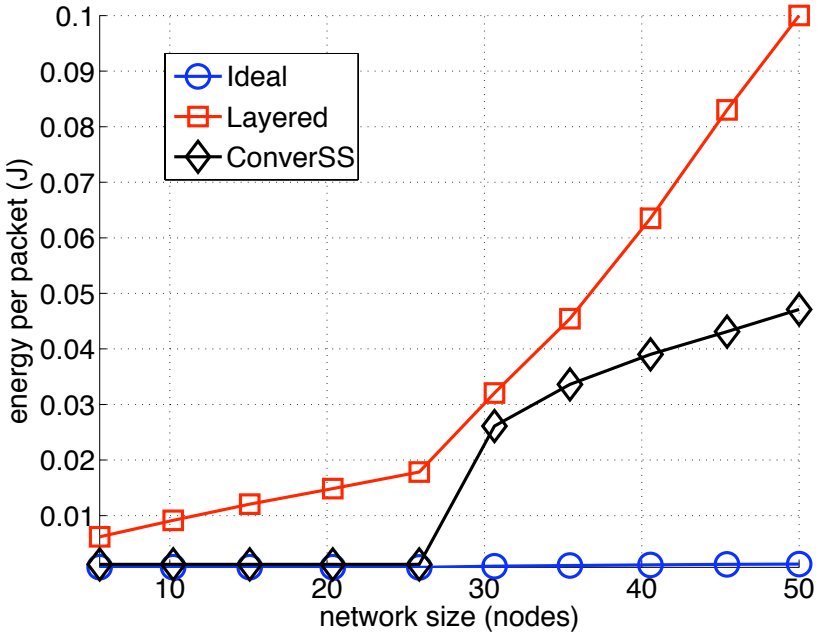


(a) Average number of hops per node.



(b) Latency per packet.

Figure 4.6: ConverSS Simulation Evaluation: Varying Network Size.



(c) Energy per node per packet.

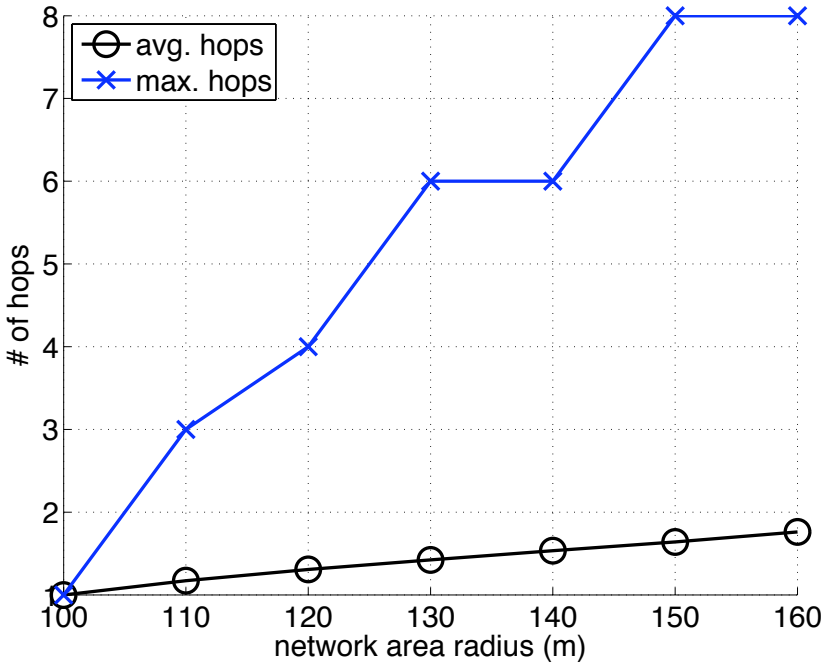
Figure 4.6: ConverSS Simulation Evaluation: Varying Network Size.

price of always listening in each others' slots, in case a route is needed.

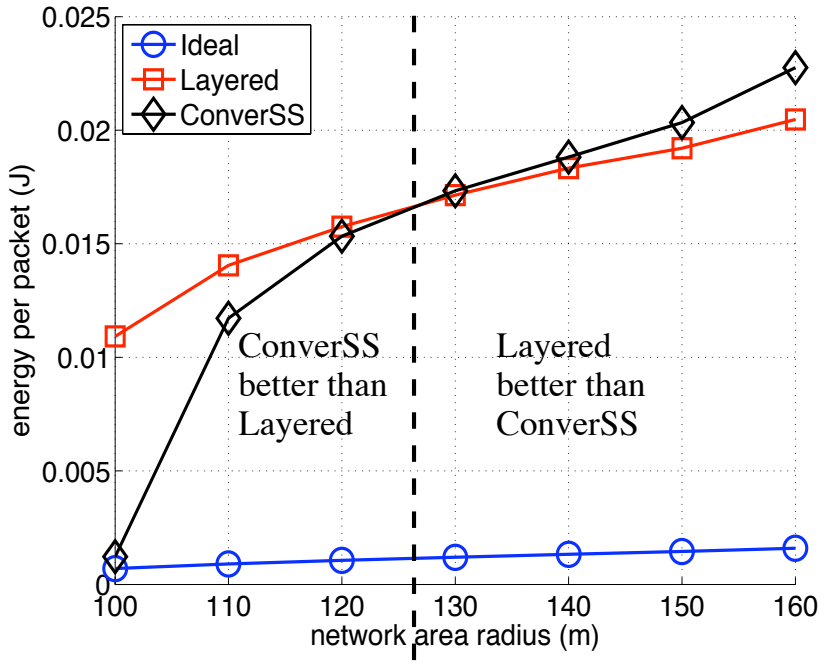
For network sizes greater than 25 nodes, we observe the performance as multi-hop routes are introduced (Fig. 4.6a). The latency increases for all protocols (Fig. 4.6b), but it is greater for *ConverSS* due to the Phase 2 frame structure, which inserts a sink slot after every node's slot. Concerning energy, *ConverSS* consumes more energy than *Ideal* due to the flooding, but it still consumes less than *Layered* (Fig. 4.6c) because only a small fraction of nodes are flooding.

4.4.2 Multi-hop Routes

For the second simulation, we study the performance when most nodes require multi-hop routes, a scenario that is atypical for these vehicle networks. Although multi-hop routes are rare for these applications, we observe that ConverSS is still equipped to handle such cases. We again fix an error-free transmission range of 100 meters, but now we fix the number of nodes at 10. We vary the node den-



(a) Average and maximum number of hops.



(b) Energy per node per packet.

Figure 4.7: ConverSS Simulation Evaluation: Multi-hop Routes.

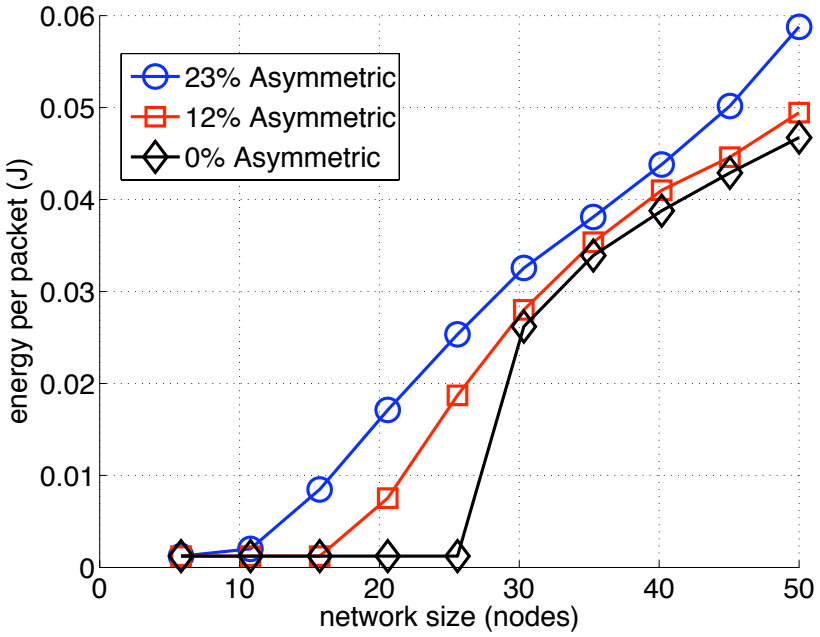
sity by varying the area over which these 10 nodes are randomly distributed. Only those deployments for which all nodes have a route to the sink were considered. As the network area grows, nodes are spread out farther over more hops, and the percentage of 1-hop nodes decreases. Fig. 4.7 plots the average and maximum hops per node and the energy consumption.

Although we have not shown it here, nodes with a route to the sink have a packet delivery rate of 100% success. We see in Fig. 4.7b that the energy consumption of *ConverSS* increases at a faster rate than *Layered*. For network area radii greater than 127 meters, *ConverSS* actually becomes the less energy-efficient choice. This is due to the inefficiency of flooding when nodes get stretched out to have longer routes (Figs. 4.7a). This simulation illustrates that ConverSS is not meant as a generic solution for all types of networks. It is not designed for networks consisting of mostly multi-hop routes, but instead, it is meant for small networks, which consist of mostly one-hop nodes. Still, ConverSS is fully capable of handling general connected networks because of the fallback mechanism.

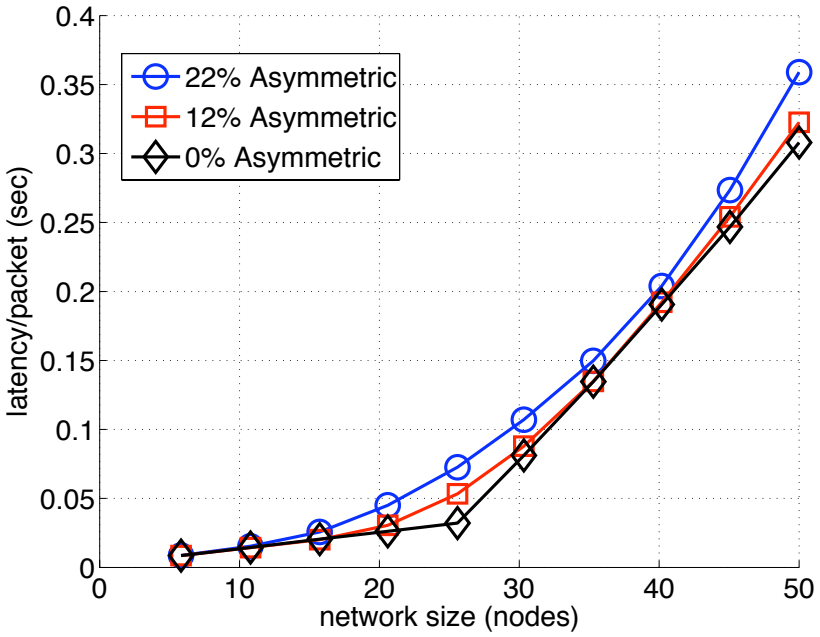
4.4.3 Link Asymmetry

Lastly, we simulated *ConverSS* for networks that contain asymmetric links. To simulate these effects, we introduced randomness to each node's transmit power and receive threshold, which varies the transmission range independently for each link in each direction. Three different degrees of asymmetry were simulated, resulting in networks for which 23%, 12%, and 0% of the links were unidirectional. We used the same network settings as in Simulation 1. The results are shown in Fig. 4.8.

Again, though not displayed here, all cases achieve 100% delivery success for nodes that have a path. From Figs. 4.8a-4.8b, we see that the latency and energy increase slightly as asymmetry increases. This is because nodes with asymmetric links are unable to hear when their transmission has been received by the sink or forwarded by another node. Therefore, nodes must retransmit those packets, and in some cases, the flooding is not controlled. Although the presence of asymmetric links incurs a slight penalty in performance, we observe that ConverSS is still



(a) Energy per node per packet.



(b) Latency per packet.

Figure 4.8: ConverSS Simulation Evaluation: Link Asymmetry.

Figure 4.9: 802.15.4-compliant TelosB mote from Crossbow, Inc.

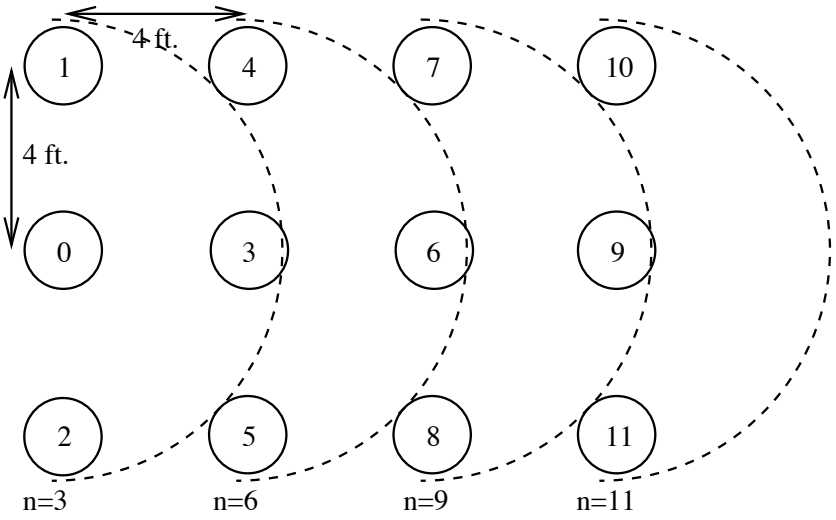
resilient to such impairments.

4.5 Experimental Evaluation

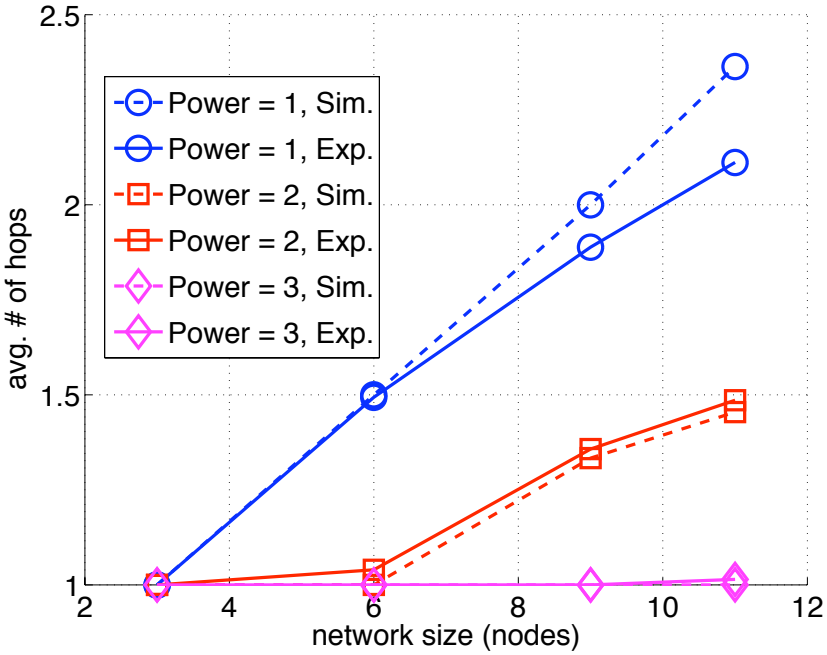
The network simulator provided an evaluation of the protocol performance that assumed an idealized model of the wireless channel. To evaluate the performance of the protocol in a real-world scenario, we implemented ConverSS in an actual network deployment using the 802.15.4-compliant TelosB motes from Crossbow, Inc. (Fig. 4.9). This test also serves as a scaled-down version of our actual UAV application. The energy consumption was calculated using the power consumption values for the ChipCon CC2420 transceiver shown in Table 4.2 [CC2]. In these experiments, the system is subject to non-ideal link behavior, such as anisotropic transmission range and packet losses due to time-dependent fading. These are the types of phenomena common to many wireless systems, and we expect similar challenges in the UAV scenario. In our results, we demonstrate that ConverSS is able to tolerate such impairments, and that it is efficient for mostly-one-hop networks. For these experiments, we have allotted 3 frames for Phase 1 and up to 6 frames for Phase 2. Any data packets that do not reach the base station within the 9-frame data arrival period expire and get dropped.

4.5.1 Various Network Topologies

In the first experiment, we studied the performance of ConverSS for different network topologies by varying the transmit power, which yielded routes of

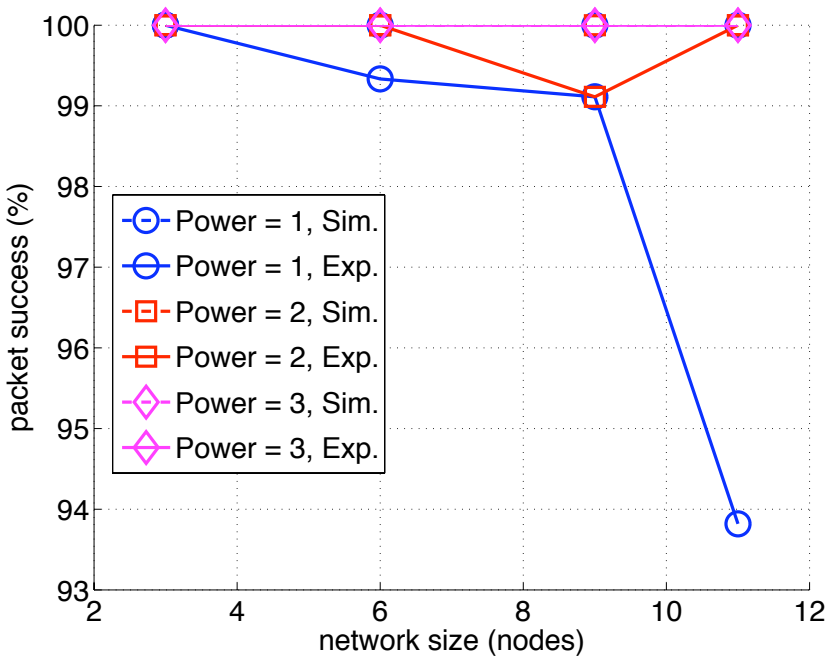


(a) Setup for Various Network Topologies.

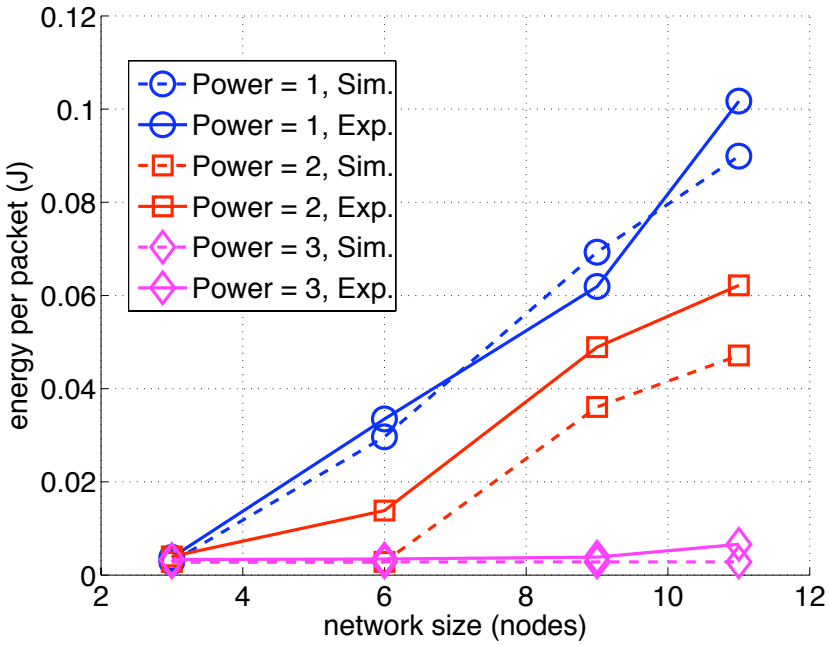


(b) Average hops per packet.

Figure 4.10: ConverSS Experimental Evaluation: Various Network Topologies.



(c) Data delivery rate.



(d) Energy per node per packet.

Figure 4.10: ConverSS Experimental Evaluation: Various Network Topologies.

Table 4.2: Power consumption of the ChipCon CC2420 IEEE 802.15.4-ready Transceiver (Transmit Power Level=3).

Mode	**Transmit**	**Receive**	**Idle**
Power Consumption	30.60 mW	67.68 mW	67.68 mW

Table 4.3: Output Transmit Powers of the ChipCon CC2420 IEEE 802.15.4-ready Transceiver.

Power Setting	**Output Power (dBm)**
3	-25
2	-28
1	-33

different lengths. As expected, we observe that the performance is better as the number of hops per route gets smaller, the scenarios for which ConverSS was designed. We deployed the nodes in a regular 3×4 grid as shown in Fig. 4.10a, and generated results for networks of various size, all sending data to node 0. For a network size = 3, only nodes 1-3 were used, for a network size of 6, nodes 1-6 were used, etc. The lowest 3 transmit power settings (out of 31) were chosen to accommodate a sufficiently small testing area. The associated output powers are listed in Table 4.3 [dPAP08]. To study how well our simulations compare with the experimental results, we have duplicated this scenario in ns-2 using error-free links. Simulated transmission ranges were chosen to match the route lengths for this particular environment, as shown in Fig. 4.10b. These ranges turned out to be 4 feet, 8 feet, and 16 feet (for Power = 1, 2, and 3, respectively).

Fig. 4.10c plots the data delivery success rate for a data arrival period of 9 frames. The data delivery rate is 100% for most scenarios. In rare cases, packets do not arrive within the given sending interval due to link failures. In Fig. 4.10d, when the network size is 3, power consumption is small for all power levels because all nodes are 1 hop away, which is when the protocol is most efficient. We observe that as the network size grows, the energy consumption increases for the lower power levels (multi-hop networks). This is because transmission ranges

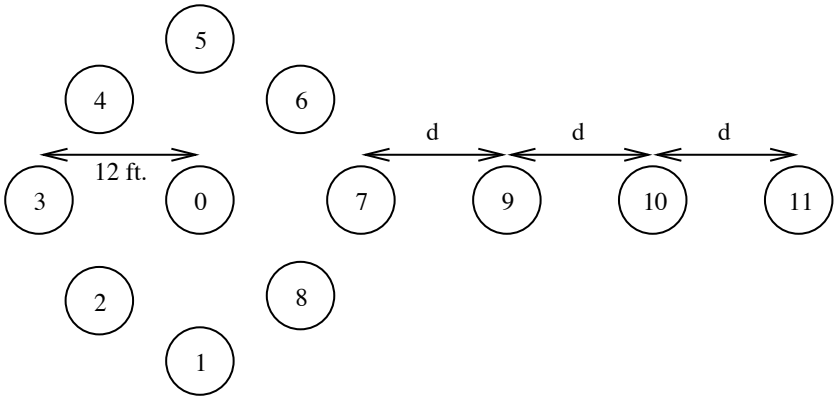
are shorter, so more hops are required to route to node 0. These conditions are rare for practical vehicle-based sensor networks since nodes will mostly be in range of the base station.

The simulation results approximate the experimental results well. In some cases, the average number of hops is lower in the experiment (Fig. 4.10b) since the actual transmission range can be greater, depending on the test environment. Also, in simulation, the power consumption is lower due to the error-free links, and 100% delivery is achieved for all cases. Still, the experimental results demonstrate good performance in spite of packet errors and link asymmetry, especially for the common scenarios.

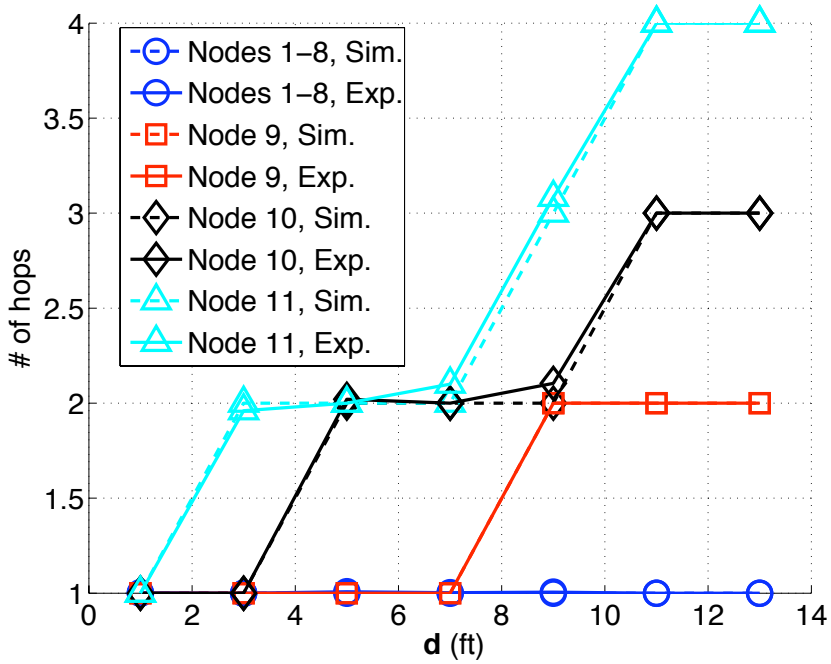
4.5.2 Multi-hop Chain of Nodes

In the second experiment, we studied the performance when most nodes are in range of the base station with a few out-of-range nodes. This scenario is sometimes found in the UAV application, in which a few planes may wander out of the main sensing area. This experiment captures the situations we expect most often for these vehicle-based sensing systems, in which up to a few nodes require multi-hop routes. The power level is fixed and set to 2, and the network is set up as shown in Fig. 4.11a. Nodes 9-11 are the out-of-range nodes, and we vary the spacing between the chain of nodes, **d**. The operation is shown to be efficient when there are fewer hops (**d** is small). We have simulated this setup in ns-2 with error-free links, and the transmission range that matches the route lengths is 19 feet (Fig. 4.11b). This range differs from that of Experiment 1 because it was conducted in a different test environment.

In Fig. 4.11, we have plotted the average number of hops, the data delivery rate, and the average energy consumption for nodes 1-8, and for nodes 9, 10, and 11 separately. Fig. 4.11c shows that the data delivery rate is best for nodes that are close to the base station, and it decreases as nodes are more hops away from the base station and as **d** gets larger(Fig. 4.11b). Just as in Experiment 1, some packets do not reach node 0 because of link failures. However, if there are there are no link failures, the data will be delivered, as the simulations demonstrate.

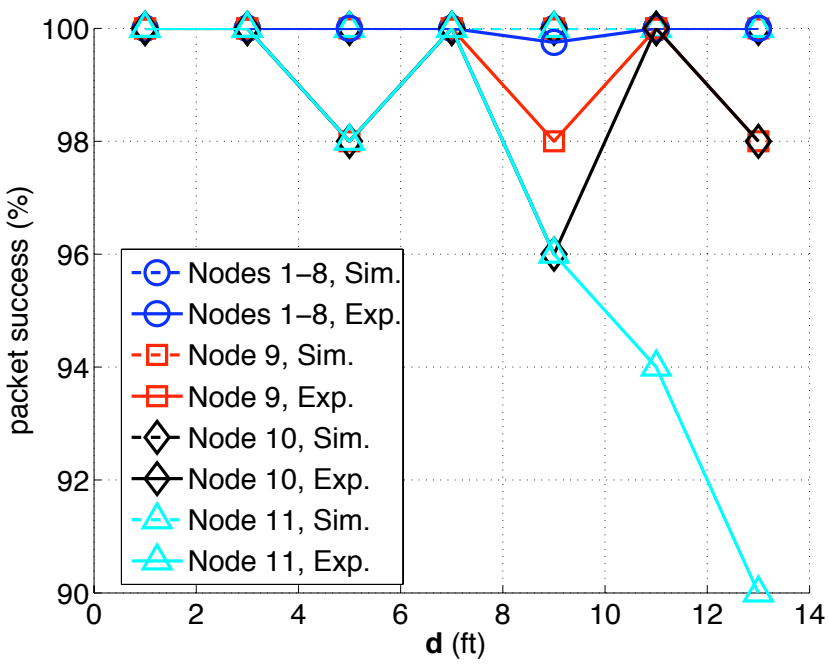


(a) Setup for Multi-hop Chain of Nodes.

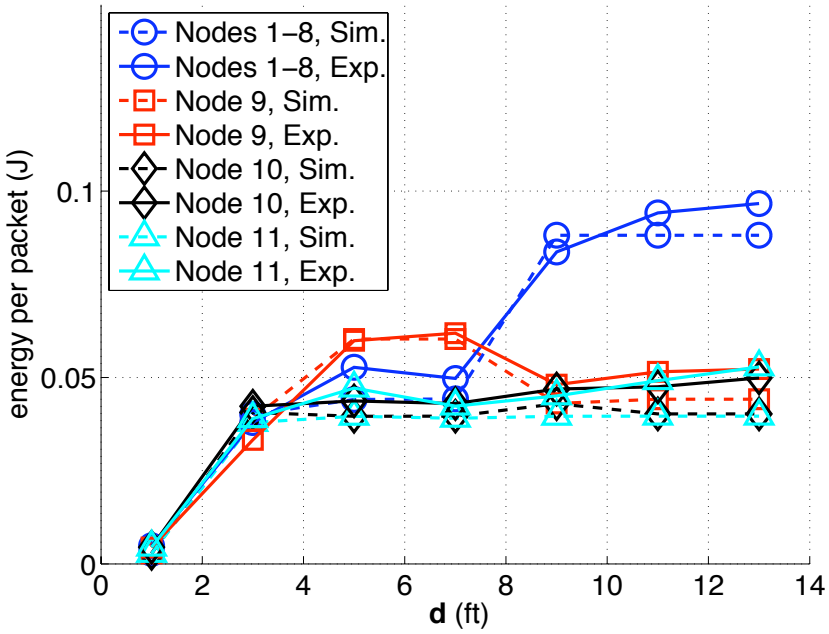


(b) Average hops per packet.

Figure 4.11: ConverSS Experimental Evaluation: Multi-hop Chain of Nodes.



(c) Data delivery rate.



(d) Energy per node per packet.

Figure 4.11: ConverSS Experimental Evaluation: Multi-hop Chain of Nodes.

The operation of ConverSS, and likewise the energy consumption of each node, is dependent on the particular network topology. The general trend is that as **d** gets larger, energy consumption increases for all nodes because of the potential of having to forward another node's data (Figs. 4.11d). For $\mathbf{d} > 9$ ft., nodes that are in 1-hop range (nodes 1-8) of the base station consume more energy largely due to the extra idle listening and receiving from the base station, which occupies every other slot in Phase 2 (Fig. 4.2).

The overall energy consumption pattern fits with our intuition of how ConverSS was designed. When nodes are in range of the base station, the protocol functions primarily as a MAC, and the energy consumption is very efficient. When some nodes are out of range, other nodes must accommodate the need for routing, and thus consume more energy, but for typical scenarios, it still outperforms layered approaches.

4.6 Related Work

There has been a lot of research into convergecast, since it is recognized as an important traffic pattern for sensor networks. Much of the focus has been on scheduling to increase throughput [LKL08] or to reduce latency [LKR04, RXMC05]. Routing is usually achieved by constructing a routing tree prior to scheduling [ZH05], but often through the use of a centralized coordinator with knowledge of the connectivity graph [TC08, AGS03]. Our approach differs in that our networks are mobile, and routes are discovered during data delivery. Directed diffusion routing [SHGE04] is data-gathering by nature, in that individual nodes request and receive data from the rest of the network. However, directed diffusion is more efficient when routes are reused, so it is not optimized for our application.

There has also been much study into routing for vehicular ad hoc networks (VANETs), but communication is typically not convergecast, and energy efficiency is not a primary focus for these automotive applications. Most routing approaches are position-based [KK00, LH04], which rely on knowledge of the positions of the source, its neighbors, and the destination. Street maps are often used to assist

in routing [BM08], which does not apply in general for our sensing applications. Some rely on flooding-based techniques for high mobility [BTD06], especially for applications such as automotive collision avoidance. For such applications, the important performance metric is latency. Wang et al. [WLH+05] proposed an intelligent flooding technique for inter-vehicle communications (IVC), designed for general data services. This technique was shown to outperform reactive routing approaches for small networks, in terms of throughput and latency.

Another protocol that is designed for these small-scale convergecast networks is TreeDMA [KS09]. The main idea of this work is that a routing tree is implicitly constructed during data delivery, using beaconing and overhearing. However, this protocol fails when packet errors and asymmetric links are introduced. ConverSS is robust to such real-world effects.

4.7 Summary

Sensor-equipped vehicle networks present an important application area in which nodes can be dispatched to a remote location to perform a task. They differ from traditional sensor networks in that they are mobile and consist of a small number of nodes. We have shown that these small-scale networks are mainly composed of nodes that are only one hop from the sink (Fig. 4.1). By designing a protocol specifically for these scenarios, we have achieved better performance than traditional layered protocols, which are designed for larger, multi-hop networks. We have introduced a two-phase, cross-layered networking approach that is robust and is optimized for these applications. For small, mostly one-hop networks, ConverSS has been shown to have energy consumption similar to an "ideal" protocol and 10 times better than a layered protocol.

4.8 ConverSS Pseudocode

Phase 1 Pseudocode

1: Wake up at start of sending interval

```
2: Set DSB all 0's
3: for all slots do
4:     if My sending slot AND have data packet then
5:         Send to Sink
6:     else if Sink's sending slot then
7:         Listen
8:         if Receive DSM then
9:             if My ACKB bit==1 then
10:                Add Sink as parent
11:                Free data packet
12:            end if
13:            Update DSB
14:            if DSB is all 1's then
15:                Sleep until the next sending interval
16:            end if
17:        end if
18:    else
19:        Sleep in slot
20:    end if
21: end for
```

Phase 2 Pseudocode

```
1: for all slots do
2:     if Beginning of frame then
3:         Schedule listening in slots of unsuccessful nodes (DSB bit == 0) and
   my parent and children node(s)
4:     end if
5:     if My sending slot AND have data packet then
6:         SEND()
7:     else if Listening scheduled in slot then
8:         LISTEN()
```

```
    else
        Sleep in slot
    end if
    if Have not sent or received for T_CD then
        Sleep until the next sending interval
    end if
end for
```

Phase 2 Send Procedure

```
procedure SEND()
    if Packet's Source Node's DSB bit==1 then
        Remove packet from queue
    else
        if Have a parent AND parent≠Packet's Source node then
            Send to parent
        else
            Broadcast data packet
        end if
        Set ACKB to all 0's
    end if
    if Sent packet was DSM OR resend threshold exceeded then
        Remove packet from queue
    end if
    if DSB is all 1's AND no packets queued then
        Sleep until the next sending interval
    end if
end procedure
```

Phase 2 Receive Procedure

```
procedure LISTEN()
```

```
2:  Listen
3:  if Receive packet for me then
4:      Set received packet's source node's ACKB bit=0
5:      if Have not yet received it before and its DSB==0 then
6:          Add received packet's source node as child
7:          Queue packet
8:      else
9:          Queue DSM (To receive packet's source node)
10:     end if
11: else if Overhear packet then
12:     if Have no parent and my ACKB bit==1 then
13:         Add received packet's source node as parent
14:         Free received data packet
15:         Update DSB
16:     else if Have a parent then
17:         if Received packet's source node is parent then
18:             if My ACKB bit==1 then
19:                 Free data packet
20:             end if
21:             Update DSB
22:         end if
23:     end if
24:     return
25: else if Receive DSM then
26:     if My ACKB bit==1 then
27:         Free received data packet
28:     end if
29:     Update DSB
30: end if
31: if DSB is all 1's AND no packets queued then
32:     if Have children then
```

```
33:             Queue DSM (broadcast)
34:         else
35:             Sleep until the next sending interval
36:         end if
37:     end if
38: end procedure
```

4.9 Acknowledgements

This chapter is, in part, a reprint of material submitted for publication in *IEEE Transactions on Mobile Computing* under the title "ConverSS: a Hybrid MAC/Routing Solution for Small-Scale, Convergecast Wireless Networks." The dissertation author was the primary researcher and author in the publication, and Curt Schurgers supervised the research.

Chapter 5

Conclusion and Future Directions

In previous chapters, we have addressed energy efficient strategies for three different sensor network applications, each of which have distinctly different connectivity properties. Real deployments have emerged that have opened up a space for energy conservation schemes focused on these different classes of WSNs. In this chapter, we review our main results and propose future research directions separately for each category.

In Chapter 2, we focused on the case for multi-hop sensor networks, with a focus on a target tracking application. Most of the focus in sensor network research has been for these multi-hop, connected networks. Sensor network applications are often designed to operate within the context of a distributed computing architecture for the purpose of energy efficiency. Our approach follows this new networking paradigm. We adapt a traditional target tracking technique within a distributed sensor networking context.

For our particular setup, we studied a network of proximity sensors, which detect an event when a node is within a given sensing range. The triggered node is responsible for the data processing and for alerting nearby nodes of an upcoming event. We gradually developed our tracking algorithm by first demonstrating the Kalman filter operation for a continuous sampling case. We then adapted the Kalman filter to a case in which data is unavailable for a number of samples. This case was handled by only running the prediction phase of the Kalman filter when no observed data was available. Finally, we adapted the Kalman filter to the proximity

sensor case, using the sensor locations as the noisy observed data, and running only the prediction phase of the Kalman filter in between the observations. Through simulation, we showed target location estimates that improved on the observed data, and the predicted error ellipses were shown to be useful for alerting nodes of upcoming events and electing the next local processor node.

In Chapter 3, we presented a power management scheme that is designed for sparse delay-tolerant networks. This type of network is distinct from the previous two in that the network is typically partitioned, and mobility is required to route data between nodes. Mobility-enabled routing is a well-studied area for DTNs, and these routing algorithms often rely on some local contextual information to make a decision on whether to hand off data messages. We use the local information available to the individual node in our power management to make a sleep/wakeup decision for the radio.

As an example of a scenario where nodes have rich local contextual information, we chose a geographic convergecast application. In this example, nodes are aware of their own geographic location, their own velocity, and the geographic location of a static sink node. This information has been used previously in a proposed routing algorithm [LCGZ05], such that nodes hand off data to other nodes whose velocities are in the direction of the sink. For our power management problem, we showed how to use this same information to put nodes to sleep. First, we used the local state to determine if a node is likely to reach the sink. Then, we used that likelihood to determine the radio sleep/awake state.

We started our design by choosing a baseline beaconing mechanism, as is common for DTNs. We devised two different sleep/wakeup strategies for nodes with data and nodes without. Since nodes without data only need to be available as relays, we chose a *threshold* on the state metric to divide the state space into sleep and awake. The threshold is chosen in such a way that the desired delivery ratio is met. Nodes with data are primarily concerned with delivering the data they already have. The strategy is to choose a *beaconing rate* at which potential relays can be discovered, such that the desired delivery ratio can still be achieved. The threshold and beaconing rates are calculated using a quantized local state space

and a Markov representation of the network dynamics. We derived the Markov model parameters using simulation and analytical derivation. We demonstrated our power management scheme in simulation and show that it outperforms existing schemes.

Our work has demonstrated the concept of using local state information to maximize energy efficiency for DTNs. What still needs to be explored is the effect of the power management on the available local information. Specifically, sleeping may reduce the accuracy of the local state information, so a power management strategy must account for keeping such accuracy within acceptable levels. For our geographic convergecast application, the information available to the power management was mostly independent of any information communicated between nodes. For other applications, there may be issues with nodes sleeping too much and as a result, lacking accurate, up-to-date information on the local state.

In Chapter 4, we focused on an example of a dense, small-scale sensor network, using a vehicle-based sensor network as the relevant application. This type of network is distinct from the multi-hop WSN in that the transmission range is on the same order as the coverage area. As a result, most nodes are within one or two transmission hops of each other. We use this fact to develop a duty-cycling solution that is optimized for our specific application. The same principle can be extended for other applications of dense, small-scale WSNs.

The specific application that our protocol is designed for is a UAV network with periodic convergecast. In this scenario, all nodes regularly send messages to the central sink. Because nodes are mostly in range of the sink, messages can be addressed directly to the sink and routing is not required. In most cases, all that is required is a MAC. However, we must account for the possibility of nodes straying from the expected coverage area, in which a multi-hop route is needed. Therefore, the overall approach we take is to have a MAC with a fallback routing scheme. We developed ConverSS, our hybrid MAC/routing solution for small-scale convergecast networks, and demonstrated its superior performance over a layered approach.

A possible extension to the dense, small-scale problem would be to mod-

ify our solution for implementation in a wider range of networking tasks. The principle of leveraging the typical small-scale topologies can be applied beyond our specific scenario. In this work, we focused on periodic convergecast, which is a very common task for these systems. As these swarming vehicles become more sophisticated, they may achieve greater autonomy and be less reliant on a base station node. The data traffic in this system would not necessarily be convergecast or periodic, so our protocol would have to be adapted for arbitrary data traffic. There is likely to be a tradeoff between energy efficiency and the ability to accommodate a greater variety of tasks.

In this dissertation, we have mapped out a broad view of sensor networks that expands on the traditional definition of WSNs. We have identified new kinds of energy-sensitive systems and have proposed strategies for energy efficiency that are designed specifically for these systems. As mobile sensor networking technologies continue to emerge, our work will have sustained applicability for the practical design of efficient sensor networks.

Bibliography

[ACFP09] Giuseppe Anastasia, Marco Contib, Mario Di Francescoa, and Andrea Passarella. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks*, 7:537–568, May 2009.

[AGS03] V. Annamalai, S. K. S. Gupta, and L. Schwiebert. On tree-based convergecasting in wireless sensor networks. In *IEEE Wireless Communications and Networking Conference*, pages 1942–1947, March 2003.

[ARE+05] Anish Arora, Rajiv Ramnath, Emre Ertin, Prasun Sinha, Sandip Bapat, Vinayak Naik, Vinod Kulathumani, Hongwei Zhang, Hui Cao, Mukumdan Sridharan, Santosh Kumar, Nick Seddon, Chris Anderson, Ted Herman, Nishank Trivedi, Chen Zhang, Mikhail Nesterenko, Romil Shah, Sandeep Kulkarni, Mahesh Aramugam, Limin Wang, Mohamed Gouda, Young ri Choi, David Culler, Prabal Dutta, Cory Sharp, Gilman Tolle, Mike Grimmer, Bill Ferriera, and Ken Parker. Exscal: Elements of an extreme scale wireless sensor network. In *Proceedings of the 11th IEEE Conference on Real-Time and Embedded Computing Systems and Applications (RTCSA '05)*, Hong Kong, August 2005.

[Bet01] Christian Bettstetter. Mobility modeling in wireless networks: Categorization, smooth movement, and border effects. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5:55–66, July 2001.

[BM08] James Bernsen and D. Manivannan. Greedy routing protocols for vehicular ad hoc networks. In *International Wireless Communications and Mobile Computing Conference, 2008 (IWCMC '08)*, 2008.

[BRS03] Richard R. Brooks, Parameswaran Ramanathan, and Akbar M. Sayeed. Distributed target classification and tracking in sensor networks. *Proceedings of the IEEE*, 91(8):1163–1171, Aug. 2003.

[BSLK01] Yaakov Bar-Shalom, X. Rong Li, and Thiagalilngam Kirubarajan.

Estimation with Applications to Tracking and Navigation: Theory, Algorithms, and Software. John Wiley & Sons, Inc., 2001.

[BTD06] Subir Biswas, Raymond Tatchikou, and Francois Dion. Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety. *IEEE Communications Magazine*, January 2006.

[CC2] Low power RF protocols – ZigBee – CC2420 – TI.com. `http://focus.ti.com/docs/prod/folders/print/cc2420.html`.

[CE04] Alberto Cerpo and Deborah Estrin. ASCENT: Adaptive Self-Configuring sEnsor Networks Topologies. *IEEE Transactions on Mobile Computing*, 3:272–285, July 2004.

[CJBM02] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *ACM Wireless Networks*, 8:481–494, September 2002.

[CK03] Chee-Yee Chong and Srikanta P. Kumar. Sensor networks: Evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, Aug. 2003.

[CS09] Bong Jun Choi and Xuemin Shen. Adaptive exponential beacon period protocol for power saving in delay tolerant networks. In *Proceedings of IEEE International Conference on Communications*, pages 1–6, Dresden, Germany, June 2009.

[CZMK03] Chee-Yee Chong, Feng Zhao, Shozo Mori, and Sri Kumar. Distributed tracking in wireless ad hoc sensor networks. *Proceedings of the Sixth International Conference of Information Fusion*, 1:431–438, 2003.

[dPAP08] Rodolfo de Paz Alberola and Dirk Pesch. AvroraZ: Extending Avrora with an IEEE 802.15.4 compliant radio chip model. In *Proc. of the 3rd ACM International Workshop on Performance Monitoring, Measurement, and Evaluation of Heterogeneous Wireless and Wired Networks (PM2HW2N 2008)*, Vancouver, British Columbia, Canada, 2008.

[FRWZ00] Elena Fasolo, Michele Rossi, Jörg Widmer, and Michele Zorzi. Wireless integrated network sensors. *Communications of the ACM*, 43:51–58, May 2000.

[HWNC05] Owen Holland, John Woods, Renzo De Nardi, and Adrian Clark. Beyond swarm intelligence: The UltraSwarm. In *Proceedings of the IEEE Swarm Intelligence Symposium (SIS2005)*, pages 217–224, Pasadena, CA, June 2005.

[JAZ05] Hyewon Jun, Mostafa H. Ammar, and Ellen W. Zegura. Power management in delay tolerant networks: A framework and knowledge-based mechanisms. In *Proceedings of IEEE Conference on Sensor and Ad Hoc Communication and Networks (SECON 2005)*, Santa Clara, CA, September 2005.

[JOW+02] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. In *Proceedings of the 10th Intl Conference on Architectural Support for Programming Languages and Operating Systems*, San Jose, CA, October 2002.

[KK00] Brad Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, 2000.

[KS09] Clement Kam and Curt Schurgers. TreeDMA: a hybrid MAC/routing solution for small-scale wireless networks. In *Proceedings of the 3rd International Workshop on Information Fusion and Dissemination in Wireless Sensor Networks (SensorFusion'09 Mobiquitous'09)*, July 2009.

[LCGZ05] Jason LeBrun, Chen-Nee Chuah, Dipak Ghosal, and Michael Zhang. Knowledge-based opportunistic forwarding in vehicular wireless ad hoc networks. In *IEEE Vehicular Technology Conference*, pages 2289–2293, 2005.

[LDS04] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *Service Assurance With Partial and Intermittent Resources*, 3126:239–254, 2004.

[LH04] Jun Luo and Jean-Pierre Hubaux. A survey of inter-vehicle communication. Technical Report IC/2004/24, EPFL, Lausanne, Switzerland, 2004.

[LKL08] Nai-Luen Lai, Chung-Ta King, and Chun-Han Lin. *Lecture Notes in Computer Science*, volume 5036/2008, chapter On Maximizing the Throughput of Convergecast in Wireless Sensor Networks, pages 396–408. Springer Berlin / Heidelberg, 2008.

[LKR04] Gang Lu, Bhaskar Krishnamachari, and Cauligi S. Raghavendra. An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, page 224, April 2004.

[LRZ03] Juan Liu, James Reich, and Feng Zhao. Collaborative in-network processing for target tracking. *EURASIP Journal on Applied Signal Processing*, 4:378–391, 2003.

[MN02] Donal McErlean and Shrikanth Narayanan. Distributed detection and tracking in sensor networks. *Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers*, 2:1174–1178, Nov. 2002.

[PK00] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43:51–58, May 2000.

[RSPS02] Vijay Raghunathan, Curt Schurgers, Sung Park, and Mani B. Srivastava. Energy-aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19:40–50, March 2002.

[RXMC05] Biao Ren, Junfeng Xiao, Jian Ma, and Shiduan Cheng. *Mobile Ad-Hoc and Sensor Networks*, volume 3794/2005, chapter An Energy-Conserving and Collision-Free MAC Protocol Based on TDMA for Wireless Sensor Networks, pages 603–612. Springer Berlin / Heidelberg, 2005.

[SAZ07] Lifeng Sang, Anish Arora, and Hongwei Zhang. On exploiting asymmetric wireless links via one-way estimation. In *Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '07)*, September 2007.

[SBS02] Eugene Shih, Paramvir Bahl, and Michael J. Sinclair. Wake on wireless: An event driven energy saving strategy for battery operated devices. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking (MOBICOM '02)*, September 2002.

[SH03] Tara Small and Zygmunt J. Haas. The shared wireless infostation model - a new ad hoc networking paradigm (or where there is a whale, there is a way. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc '03)*, pages 233–244, Annapolis, MD, June 2003.

[SH05] Tara Small and Zygmunt J. Haas. Resource and performance tradeoffs in delay-tolerant wireless networks. In *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-Tolerant Networking*, pages 260–267, Philadelphia, PA, August 2005.

[SHGE04] Fabio Silva, John Heidemann, Ramesh Govindan, and Deborah Estrin. Directed diffusion. Technical Report ISI-TR-2004-586, USC/Information Sciences Institute, Los Angeles, CA, January 2004.

[SPR04] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Single-copy routing in intermittently connected mobile networks. In *First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, pages 235–244, Santa Clara, CA, October 2004.

[SPR05] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay Tolerant Networking*, pages 252–259, Philadelphia, PA, August 2005.

[SPR06] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi Raghavendra. Performance analysis of mobility-assisted routing. In *Proceedings of ACM MOBIHOC*, pages 49–60, Florence, Italy, May 2006.

[STGS02] Curt Schurgers, Vlasios Tsiatsis, Saurabh Ganeriwal, and Mani B. Srivastava. Optimizing sensor networks in the energy-latency-density design space. *IEEE Transactions on Mobile Computing*, 1:70–80, January/March 2002.

[STS02] Curt Schurgers, Vlasios Tsiatsis, and Mani B. Srivastava. Stem: Topology management for energy efficient sensor networks. In *IEEE Aerospace Conference '02*, pages 1099–1108, Big Sky, MT, March 2002.

[TC08] Hua-Wen Tsai and Tzung-Shi Chen. Minimal time and conflict-free schedule for convergecast in wireless networks. In *IEEE International Conference on Communications*, May 2008.

[VB00] Amin Vahdat and David Becker. Epidemic routing for partially-connected ad hoc networks. Technical Report CS-2000-06, Duke University, Durham, NC, April 2000.

[vDL03] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (Sensys'03)*, pages 171–180, Los Angeles, CA, November 2003.

[WALW+06] Geoffrey Werner-Allen, Konrad Lorincz, Matt Welsh, Omar Marcillo, Jeff Johnson, Mario Ruiz, and Jonathan Lees. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 12:18–25, March/April 2006.

[WLH+05] S.Y. Wang, C.C. Lin, Y.W. Hwang, K.C. Tao, and C.L. Chou. A practical routing protocol for vehicle-formed mobile ad hoc networks on the roads. In *Proceedings of the 8th International IEEE Conference on Intelligent Transportation Systems*, 2005.

[WW06] Yu Wang and Hongyi Wu. Dft-msn: The delay/fault-tolerant mobile sensor network for pervasive information gathering. In *Proceedings of the 25th IEEE International Conference on Computer Communications*, pages 1–12, Barcelona, Spain, April 2006.

[XCC07] Yong Xi, M. Chuah, and K. Chang. Performance evaluation of a power management scheme for disruption tolerant network. *Mobile Networks and Applications*, pages 370–380, September 2007.

[XHE01] Ya Xu, John Heidemann, and Deborah Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pages 70–84, Rome, Italy, 2001.

[YHE04] Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 12:493–506, June 2004.

[ZC98] Chenxi Zhu and M. Scott Corson. A five-phase reservation protocol (FPRP) for mobile ad hoc networks. In *Proceedings of IEEE INFOCOM*, pages 322–331, San Francisco, CA, April 1998.

[ZH05] Ying Zhang and Qingfeng Huang. Coordinated convergecast in wireless sensor networks. In *Military Communications Conference (MILCOM 2005)*, 2005.

[ZHKS06] Gang Zhou, Tian He, Sudha Krishnamurthy, and John A. Stankovic. Models and solutions for radio irregularity in wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, May 2006.

[ZR03] Michele Zorzi and Ramesh R. Rao. Geographic random forwarding (geraf) for ad hoc and sensor networks: Energy and latency performance. *IEEE Transactions on Mobile Computing*, 2:349–365, October/December 2003.