Situation-Aware Optimizations in Challenged Networks

Yong Wang

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
By the Department of
Computer Science

September 2007

Abstract

Challenged networks refer to networks with unconventional difficulties, such as intermittent connectivity, large delay, and others. Their unique communication characteristics create new challenges for the research community and demand novel solutions to achieve efficient routing and maintain existing network services.

In this dissertation, I explore new optimizations to combat these challenges under the unifying theme of achieving situation awareness. In particular, I study four categories of networks that contain a range of disruptions: highly varying mobility, lossy radio links, opportunistic connectivity, and intermittent connectivity.

The first category consists of networks with a varying mobility pattern found in many challenged mobile networks. I propose a model-based approach to capture mobility phase changes in order to maintain efficient routing. When evaluated using a real-world mobility trace, our approach leads to an improvement of up to 120% in packet delivery rate.

The second category consists of networks with lossy links. Since data collection is frequently disrupted by the difficulty of identifying good links, I use supervised learning to maintain accurate link quality information under heavy traffic load when traditional approaches fail. Our approach yields improvements of up to 300% when evaluated on a real-world sensor network testbed.

The third category consists of networks with unpredictable mobility in which data can only be forwarded in a store-and-forward fashion. Existing approaches depend heavily on mobility prediction, which is difficult though, if not impossible. I use erasure coding to forward coded data to more contacts to combat inaccurate predictions. Simulation results show that our approach has a smaller worst case delay compared to other state-of-the-art algorithms.

The fourth category consists of static sensor networks with intermittent connectivity. In such networks, energy saving opportunities arise during network disconnection. I propose a new transport protocol to leverage such opportunities that yields significant idle energy savings compared to existing approaches.

Overall, this dissertation investigates a range of challenged networks and propose a set of techniques to enable situation-awareness in order to achieve high routing performance and energy efficiency. The outcomes reveal high potential for situation-aware techniques and provide new perspectives on optimizations in challenged networks.

Acknowledgments

I consider myself immensely fortunate to have pursued my Ph.D. at Princeton University. To have Margaret Martonosi and Li-Shiuan Peh as my Ph.D. advisors has been one of the most rewarding experiences of my academic career. From the moment that I joined the ZebraNet project to the end of my thesis work, they have devoted countless hours to helping me in all aspects of my work, be it to discuss my questions and concerns, or to share their insights on research, grad school, or job hunting. They have given me the necessary freedom to work on topics of my choice. Tapping knowledge from two great minds leads me to a broader understanding of research problems, and inspired new perspectives and directions. I am very grateful to their firm belief and constant support, without which this dissertation would not have been possible.

I would also like to thank the rest of my thesis committee, Jennifer Rexford, Larry Peterson, and Kai Li, for their insightful comments on this work. Their feedback has greatly improved this dissertation. I also wish to express my gratitude to Sushant Jain, Kevin Fall, and Chieh-Yih Wan. I gained tremendous knowledge and experience working with them.

I am deeply grateful to many people who helped me along the journey, in ways both big and small. It has been a pleasure to get to know and to work with all of them. I thank Professors Brain Kernighan and Vivek Pai for setting great examples for pedagogy. I thank my colleagues in the Sensor Network Operation (SNO) group at Intel Research, in particular Mark Yarvis and Chieh-Yih Wan, for the opportunity to work with leading engineers in the field on real-world applications and their mentoring. I thank Yang Guo from Thomson Research for hosting me as a summer intern to work on a cutting-edge sensor network system. I thank my present and former colleagues in the MRMgroup (Zhigang Hu, Russ Joseph, Hidekazu Oki, Philo Juang, Fen Xie, Eric Chi, Gilberto Contreras, James Donald, Maria Kazandjieva, Vincent Lenders, Ting Liu, Canturk Isci, Christopher Sadler, Pei Zhang, Abhishek Bhattacharjee, Manos Koukoumidis, and Carole-Jean Wu) and my colleagues in Professor's Peh's group for the countless discussions, writing revisions, and practice talks. Working with them has truly benefited my work and made my graduate years a lot more fun.

I thank my fellow students both inside and outside Princeton, in particular Ge Wang, Qiang Wu, and Limin Jia, Shane Eisenman of Columbia University, and Robert G Taylor of the University of Manchester, for your friendship and help. Special thanks to the staff in the CS and EE departments, in particular Melissa Lawson and Stacey Weber, for all their help with anything and everything administrative over my six years at Princeton. Melissa has also spent extensive time in holding English practice sessions with me during my beginning years at Princeton, through which I have learned a great deal.

Last but not least, I thank my family, my parents and parents-in-law, for their endless love, support, and care. They gave me the drive to accomplish my goals as I ventured along the path. A big "thank you" to my elder brother, Jun, and his family. Finally, I would like to thank my wife, Weiwei, who has shared every moment with me over the years and has helped me in many ways that I can hardly recount here. She made these Ph.D. years enjoyable and memorable.

This work was supported in part by National Science Foundation grants (CNS-0205214, CCF-0086031, CMS-0407555/540804-8001, CNS-0614949, and CCF-0237540), and grants from Intel Corporation.

Contents

List of Figures

Chapter 1

Introduction

The main themes of this dissertation are to study the performance problems in emerging challenged wireless and mobile networks and to investigate network- and upper-layer solutions for communication optimizations.

Emerging over the last few years, challenged networks are becoming a major part of the communication technology landscape, due to the evolution of wireless technologies and the increase in application demands. Although there is no universal definition, challenged networks generally refer to those characterized by challenges such as disconnection and long delays. They have been used to describe many sub-areas of wireless and mobile networks, such as wireless sensor networks [31, 32], delay/disruption tolerant networks (DTNs) [33], and opportunistic networks [89]. Applications running on such networks include wildlife tracking [58], IT support to developing regions [10, 11], and habitat monitoring [116]. They have revolutionized the way communication occurs and they provide new opportunities to observe and interact with unexplored areas. However, despite their promises in extending communication to broader regions, these networks are often performance-challenged, due to the limited infrastructure support and the harsh environment where they are usually deployed. They are designed to tolerate a wide range of networking challenges, such as intermittent connectivity, unreliable link transmissions, long propagation delays, and the absence of end-to-end path.

First, we will motivate this dissertation with examples of challenged networks and their typical application scenarios. We will then highlight the difficulties faced in providing efficient communication in such networks.

1.1 Challenged Networks: Application Scenarios

One important application domain in challenged networks is wireless sensor networks, which have witnessed tremendous growth in recent years. Sensor networks extend human vision and understanding by allowing sensing data from the physical world to be automatically and reliably collected. The growth of sensor networks research has led to many useful real-world deployments, such as environmental monitoring [116, 131], wildlife tracking [58] and structure monitoring [138].

Hereafter, our focus in this dissertation is on challenged sensor networks. This is because they represent a large category of challenged networks and possess a variety of challenges encountered in reality. Furthermore, wireless sensor networks are often characterized by severe energy and resource constraints, which make their networking designs even more difficult compared to those for networks with more powerful nodes and less constraints, such as cellular phones, PDAs, and laptops. Therefore, the solutions proposed and the insights gained in this dissertation are relevant to other challenged networks and would apply as well.

Usually, a sensor node comprises sensing, computing, wireless communication, and storage components. They are used to measure the ambient environmental conditions and then transport collected information to an external base station where data can be processed to reveal characteristics of the environment or the objects being sensed. However, these sensors are extremely constrained in their energy, computing, and storage capacities, which makes it very difficult to maintain high performance routing.

Wireless sensor networks can be either static or mobile. Both types of networks are performance-challenged, though they possess very different characteristics.

Static sensor networks. Static sensor networks have been successfully used in many real world deployments [67, 116, 119, 131, 138]. Usually, these static networks contain hundreds of sensor nodes that are stationary once deployed. The sensor nodes are normally deployed in high density and work cooperatively for a long period of time unattended. Multihop routing is usually adopted to reduce energy consumption since direct long-distance communication between a sensor and the base station is only possible using prohibitively high transmission power. Using multihop communication may also help to mitigate the interference between concurrent radio transmissions. Hence, multihop routing is widely used in many static sensor networks.

Collection is a fundamental component of many sensor network applications. As a result, the first generation of sensor network deployments focused primarily on data collection [76, 115]. Since users are usually interested in data or events that are covered by many sensors, most data collection mechanisms developed today involve transmitting data from many sensors to one or more sinks that could either be statically deployed or continuously moving (if the sink is a user who moves around the area covered by the network). The dominant communication pattern, therefore, is a many-to-one tree-based routing, in which multiple data collection trees rooted at the data sinks are created, with various forms of aggregation along the collection paths. Data collection routing is also referred to as convergecast in the literature, which is used to emphasize the direction of data flows compared to broadcast.

Mobile sensor networks. Compared to static sensor networks, mobile sensor networks have not been studied as extensively. They emerged as a significant new research field over the last few years as more applications in sensor networks have mobility as an inherent component. Due to the diverse operation environments, hardware configurations, and design goals in mobile sensor networks [7, 13, 48, 58], there are no universally accepted platforms and system-level solutions in these networks. In particular, the diverse nature of mobility in such networks makes it very hard to have an efficient protocol that works across different mobilities.

Applications of mobile sensor networks include wildlife tracking [58], Pocket Switched Networks [47], and participatory sensing [13]. In these networks, nodes move under the control of the environment, the object on which the node is attached, or the node itself. They can cover a larger geographic area and encompass a larger range of data with potentially fewer nodes than stationary networks. This is particularly important for applications with logistical concerns, including monetary cost, that make large scale deployment infeasible.

ZebraNet is a mobile sensor network that targets wildlife tracking across large regions with no communications infrastructure [58]. It is in essence a mobile ad hoc network (MANET) of resource-constrained sensor nodes and intermittent connectivity. In ZebraNet, nodes move throughout an environment to collect information about their surroundings. Periodically, logged GPS data is aggregated to the base station, which is also constantly moving to increase the probability of data homing success. The problems posed by the ZebraNet project are characteristic of many other mobile challenged networks and we classify them into three categories: (1) sparse and intermittent con-

Table 1.1: Comparison between mobile and static sensor networks.

	Static	Mobile
Mobility	no	yes
Density	high	low
Connectivity	good	intermittent
Routing	multihop	store-and-forward, multihop
Coverage	small	large
Link reliability	low	low
Cost per node	low	high
Energy supply	battery or environmental energy	battery or environmental energy

nectivity, (2) unpredictable and highly varying node movements, and (3) limited energy budgets and link bandwidth.

The different challenges in static and mobile sensor networks lead to different tradeoffs and designs as to how information should be collected and disseminated over the network. In static sensor networks, the dynamics come from environmental conditions that affect the radio link quality. Otherwise, the network operates in a relatively predictable way. With mobile nodes, however, data collection and dissemination becomes more complex because the dynamics of node mobility are usually unpredictable. Furthermore, some mobile sensor networks are sparsely connected with only intermittent connectivity. Therefore, routing in such networks requires novel solutions that can efficiently combat these difficulties. Table 1.1 summarizes the major differences between static sensor networks and mobile sensor networks, with each filling a unique niche that supports different application needs in sensor networks.

1.2 Communication Challenges

Routing is one of the most fundamental problems in challenged networks, which involves two general objectives:

1. **Efficiency:** The bandwidth and energy budget should be used efficiently to achieve high performance without depleting network resources before the targeted network lifetime.

2. **High performance:** Data yield, network lifetime, and latency are some of the most important performance metrics for routing protocols. Given the stringent resource constraints, it is very challenging to achieve these metrics at the same time. Often, such performance goals are contradictory to each other.

Achieving these objectives in challenged networks requires us to revisit many long-standing solutions, because these networks exhibit a set of communication problems that are fundamentally different from those found in traditional networks. They have led people to move away from traditional designs to solutions that do not reply upon end-to-end connectivity or reliable links. In the following, we discuss some of the key challenges with their implications to routing designs in such networks.

1.2.1 Dynamic Radio Frequency (RF) Environments

One major difference between wireless networks and wired networks is in the physical layer technology. Radio communication performance is determined largely by the signal-to-noise ratio and many factors may influence this, such as environmental noise, multipath fading, interference [111] and coexistence of other networks. Therefore, radio transmission exhibits very complex behavior that cannot be easily captured and characterized using simple models. Many empirical studies [21, 22, 112, 113, 145] have confirmed that the RF environments using low-power wireless radio transceivers are highly time-varying. These studies have guided the design decisions for challenged network routing protocols: To maintain routing efficiency, high quality paths need to be established and new methods need to be developed to track link quality dynamics.

1.2.2 Resource Constraints

Challenged networks are extremely constrained by their energy supply as nodes are often powered by battery, or by harvesting energy. Therefore, energy is one primary design constraint in challenged networks. On the other hand, such networks are usually expected to run for long periods of time, from several months to even a few years. These two contradictory goals make energy-efficient routing design a very challenging task. Since energy consumption is strongly related to node activity, the node hardware should be turned off most of the time and be activated only when necessary to accommodate the stringent energy budget; this is referred to as duty cycling. For high data rate sensing applications [138], however, simply reducing the duty cycles of sensor nodes will not work since the nodes need to be frequently active for data sampling and sensing; new solutions are called for in such scenarios.

Other than energy constraints, nodes are also constrained by their processing capabilities, storage capacities and communication bandwidth. For example, the MicaZ sensor node has only an 8MHz 8-bit micro-controller and a 4KB RAM. High-end sensor nodes, such as the Imote2 [137], are less constrained compared to early generations of sensors: It has a 416 MHz 32-bit processor and 256KB SRAM, 32MB FLASH and 32MB SDRAM. However, depending on the application requirements, such nodes may still face difficulties meeting the performance requirement. For instance, even though high-capacity drives may become commonplace in the near future [78], applications such as camera sensor networks still need to transmit and store large amount of image data, which makes storage still a top design concern.

Another problem lies with link bandwidth, which is very limited given that most sensor nodes use low-power radio transceivers for communication. One of the mostly widely used radio chips — the Chipcon CC2420 [8], is a low-cost radio transceiver designed specifically for low-power, low-voltage RF applications in the 2.4 GHz unlicensed ISM band. It implements the ZigBee/IEEE 802.15.4 standard and has a maximum data rate of 250Kbps. Due to the high error rates of radio transmissions in real-world environments, very low channel utilization can be achieved in practice [37]. This poses serious problems and limits the achievable data delivery rate, if the periods of communication contact are short and opportunistic, as in many challenged networks [89].

1.2.3 Operating Environment

Network size in challenged networks is another key design parameter. Although the ZebraNet node [144] is a sensor node with high compute capability and radio range, only tens of nodes are deployed due to logistical concerns. Another example is sensor networks deployed in the polar regions where large-scale deployment is impossible due to the severe environmental conditions. In such networks, sparse connectivity is the norm. Since nodes can barely find neighbors to forward data, end-to-end connectivity may not be guaranteed. As a result, networks may be partitioned for extended periods of time.

On the other hand, for dense networks with thousands of nodes, scalability becomes an issue. The resource-constrained nodes cannot operate efficiently using traditional point-to-point routing [6, 36] with so many nodes in the network. One particular challenge is to manage the routing table for so many nodes, given severe memory constraints.

Additionally, challenged networks are often deployed in harsh environments with no infrastructure support. In many mobile applications, node mobility cannot be controlled and network topology is constantly changing in unpredictable ways. As a result, these networks are faced with unusual situations that cannot be easily managed using traditional methods.

In summary, all these new characteristics make a compelling case for reconsidering the fundamental issues in networking designs for challenged networks. We will focus on challenged sensor networks in this dissertation.

1.3 Situation-Awareness

Most of the networking protocols widely used today make implicit assumptions about the network, such as fixed network topology, reliable links, and continuous end-to-end connectivity. Since disruptions break these implicit assumptions, many traditional networking protocols are rendered impractical, and their performance suffers significantly from a lack of knowledge about network conditions or other useful information related to the disruptions. The tradeoffs in terms of performance, energy usage, and response time in a dynamic network vary greatly, depending on system requirements, available resources, and economic motives. Exposing such situation knowledge can help protocols make fine-tuned, informed decisions, and sustain high performance with low cost. It is desirable, therefore, to make the protocols "cognitive" in an environment with extreme situations being the norm. By being "cognitive", we refer to the capability to perceive current network conditions and make adjustment and actions based on the overarching performance objective.

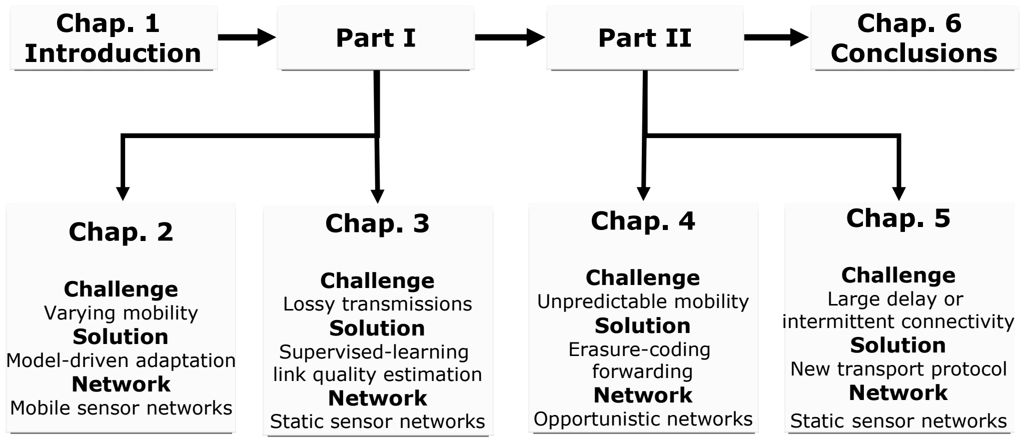


Figure 1.1: Roadmap of the dissertation.

Unfortunately, most protocols developed today do not have such cognitive capabilities. They are designed for the worst or average scenarios, or assume pre-existing situation knowledge, or tackle such challenges without considering constraints such as energy and storage. Such efforts result in protocols that work only for a pre-defined set of conditions.

We propose situation-awareness as a means for achieving such cognitive capabilities. Situation, in our definition, refers to network parameters, performance measures, or root causes of disruptions that are related to the challenges we discussed previously in this chapter. We will demonstrate in this dissertation that situation-awareness can bring significant performance benefits to challenged networks. In particular, this dissertation contributes a suite of solutions for communication optimizations based on situation awareness in performance-challenged and resource-constrained networks. Our suite consists of two major parts: providing situation-awareness and novel energy-efficient routing design, as illustrated in Figure 1.1.

In general, a situation-aware protocol consists of two core components: the networking stack and the situation information. The first part of this dissertation focuses on providing situation information to existing protocols as knowing more information can lead to more accurate routing decisions. However, this approach may not always produce good performance. For example, a protocol assuming end-to-end connectivity cannot be easily improved to accommodate frequent disconnections. Under such circumstances, we cannot optimize routing performance by simply adapting traditional routing protocols with more information. We investigate solutions for such cases in the second part of this dissertation.

1.4 Dissertation and Contributions

In this dissertation, we analyze the problems imposed by these emerging networks and present solutions that improve the overall routing performance and maintain energy efficiency via situation awareness. Due to the diversity of challenges, there is no cure-all that combats all the problems. Instead, we develop a suite of techniques to address them, under the unifying theme of achieving *situation awareness*.

1.4.1 Part I: Providing Situation-Awareness

In this part, we study two categories of challenges: highly varying mobility and lossy radio transmissions.

Varying mobility patterns in mobile sensor networks. A varying mobility pattern is typical in many emerging networks, an example of which is ZebraNet. It refers to a type of mobility in which nodes move in phases with different characteristics. We propose an analytical model to capture mobility phase changes based on past mobility traces. This model is used to drive adaptive routing decisions so that the routing strategies selected will fit the recently-observed mobility patterns. When evaluated using a mobility trace synthesized from the ZebraNet deployment data, the adaptive approach leads to an improvement of up to 120% in packet delivery rate.

Lossy transmissions in static sensor networks. High-density wireless sensor networks have been used widely for collecting environmental data in applications such as structure monitoring. Wireless sensor notworks are different from wired networks in that the link quality fluctuates greatly as a consequence of interference and propagation dynamics. Therefore, developing efficient routing in wireless sensor networks requires the establishment of high quality paths, which in turn entails accurate knowledge of link quality. Existing link quality estimation methods, however, fail in the presence of congestion and interferences since they rely on snooping data traffic [27, 136]. We use supervised learning techniques to address such limitations and pinpoint the best links without depending on data traffic. It works by selecting links based on knowledge of link quality learned from past observations during a training phase; it adds no extra overhead to the routing process. This approach is more adaptive than model-based methods because it requires minimal expert knowledge and is able to respond to changes. Our learning-enabled technique yields a performance improvement of up to 300% under heavy load, compared to the state-of-the-art approaches.

1.4.2 Part II: Novel Routing Designs

In part II, we focus on new routing designs for situations with no existing solutions, as illustrated in Figure 1.1. In contrast to examples in Part I where the solutions to a specific situation are known *a priori*, situations discussed in this part do not have existing working solutions, which leads us to explore new networking designs. Along this line of research, we develop several novel solutions to achieve efficient communication with low energy usage.

Unpredictable mobility in opportunistic networks. In opportunistic networks, end-to-end connectivity is either not always available or does not exist at all. Therefore, data is best transmitted by a store-and-forward approach. This is true for a range of networks, such as ZebraNet, Pocket Switched Networks and vehicular networks. Most existing approaches rely on mobility prediction and message replication to meet their energy budgets. Their effectiveness depends heavily on the accuracy of their mobility prediction schemes. However, making high accuracy mobility predictions is difficult in opportunistic networks. Incorrect predictions lead to poor performance and a waste of energy. To address this, we have proposed (in collaborative research) a method that forwards erasure-coded blocks, instead of replicated messages. By forwarding code blocks to more neighbors, the chances of message delivery and the worst case delivery delay are significantly improved, with the forwarding overhead kept low. Our simulation results show that the coding-based algorithm achieves a significantly smaller worst case message delay (from 60% to 70% less), compared to four other replication-based forwarding algorithms.

Intermittent connectivity in delay-tolerance, static sensor networks. New energy optimization opportunities arise in such networks due to either a relaxed latency requirement or the long time a node spends in idle waiting. We propose a new transport protocol that leverages such new opportunities and the relatively low cost of storage in current sensor devices to improve idle energy efficiency. Experimental results show that the new transport protocol yields up to 50% energy savings for a typical challenged environment, compared to existing approaches.

Overall, this dissertation investigates a range of challenged networks and proposes techniques to address new challenges with situation-awareness. It provides a roadmap to effective routing optimizations in challenged networks and provides new perspectives on their system design.

The major contributions in this dissertation are summarized as follows. First, we investigate a range of challenges that cover several key areas of challenged networks, including wireless mobile and static sensor networks, opportunistic networks, and DTNs. The idea of providing and leveraging situation information as a means to improve routing efficiency and performance in challenged networks is proposed and systematically studied in this dissertation. Second, we develop a set of techniques to provide and leverage situation information for real-world applications. These techniques are general enough to be applicable to other related problem domains. Third, our evaluations are based on either testbed implementations if conditions allow, or simulations served with real traffic and mobility. By successfully factoring real-world issues into our study, it has offered a unique perspective into communication performance optimizations in challenged networks.

1.5 Dissertation Organization

The remaining chapters of this dissertation present the major accomplishments of this research work, which are organized into two main parts: issues with providing situation awareness and issues with exploiting such exposed information. The two parts are closely related and are both integral to the overall theme of this dissertation, as they represent the two core components of any situation-aware protocol. Other than this, each chapter is relatively self-contained.

In particular, the rest of this dissertation is organized as follows. Chapter 2 presents a model-based routing framework that captures mobility characteristics in a mobile sensor network. Chapter 3 presents a solution to use supervised learning for link quality estimation in static sensor networks with lossy RF environment. These two chapters comprise "Part I" and are examples of approaches to provide situation awareness in challenged networks. Chapters 4 and 5 focus on designing new protocols for networks in which no established solutions exist or work well. By exploiting situation information, we propose totally new approaches that suit better for the targeted scenarios. In particular, Chapter 4 presents a novel routing protocol that uses erasure coding for data forwarding to cope with unpredictable mobility in opportunistic networks. Chapter 5 presents a new transport protocol for sensor networks with delay tolerance. It saves significant idle energy by removing the requirement for end-to-end connectivity between communicating pairs. Finally, we conclude in Chapter 6 with a discussion of directions for future work.

Chapter 2

Techniques for Coping with Varying Mobility

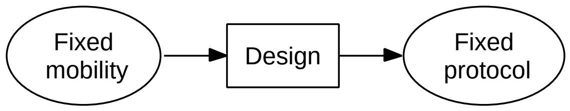
2.1 Problems and Solution Overview

The dynamics of mobile networks make efficient protocol design extremely challenging as mobility causes network topology to constantly change in unpredictable ways. Many emerging applications [30] have atypical mobility patterns such as one that alternates between highly mobile and very static movements. Since routing is governed by complex interactions between node mobility and protocol behavior, small changes in either of them may have significant impact on the overall routing performance. To maintain routing performance under varying mobility, a routing protocol needs to *adjust its behavior on-the-fly* to adapt to mobility dynamics. However, previous studies mainly focus on typical mobilities, with key routing components hand-tuned for expected mobility patterns and hard-coded ahead of time [50]. As a result, they cannot adapt to mobility dynamics in order to maintain high performance.

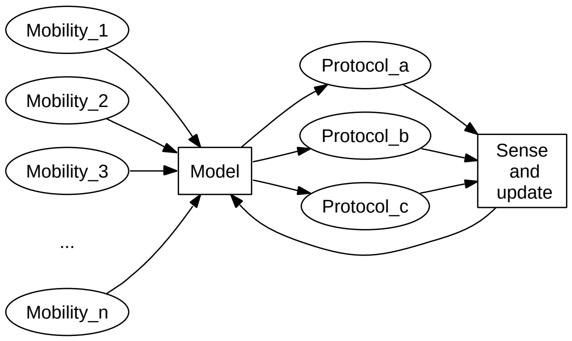
To address such inefficiencies, we propose a new component for the networking stack that provides cognitive capabilities by notifying the protocol of mobility changes. By decoupling routing decisions from mobility, we can achieve situation-awareness by dynamically enabling different routing strategies based on mobility changes. In particular, we study the Dynamic Source Routing (DSR) protocol where the effectiveness of route caching is of critical importance [45, 77]. In this context, the problem narrows down to understanding how node mobility impacts route cache access behavior and how to adjust the caching strategy to cope with mobility phase shifts.

Understanding mobility is difficult in deployed systems because measures of mobility are sometimes difficult to collect at run-time. Simulation can be used to collect such measures. However, simulation speed can be a significant problem when applied to such scenarios. We performed simulations of a 50-node mobile network for 1000 seconds on a machine with 2.2GHz Pentium 4 processor and 512MB RAM under different mobility scenarios. They each took 15 minutes to 1 hour to complete. Even worse, to explore the design space, we may need to run such simulations many times.

To overcome such shortcomings, we take an analytical approach and develop a route cache model for DSR to capture the access behavior of its route cache. A route cache access has three states: hit, miss, and false hit (wherein the route stored in the cache no longer exists due to node mobility). A false hit leads to extra processing time, network bandwidth waste and even packet drops. The model is a Discrete Time Markov Chain (DTMC) model. It accepts as input metrics collected during protocol running time and outputs various cache access rates. As node mobility and protocol behavior are both incorporated into the model, the overall routing performance is projected as a function of both of them. Hence, it can be used to “sense” the underlying mobility dynamics and help make timely decisions. The model is simple enough to be used dynamically in real-



(a) Previous approach.



(b) Our model-driven approach.

Figure 2.1: A model-driven adaptive routing optimization infrastructure. Our approach can be applied to different protocols with each protocol customized on-the-fly to mobility changes.

world settings. We also present a feedback-based optimization infrastructure that uses the model outputs to adjust the caching strategy of DSR on-the-fly.

Our approach is best understood in the context of a specific example, as shown in Figure 2.1(b). Consider the ZebraNet mobility, in which nodes move throughout the environment to collect information about their surroundings. Periodically, logged data is aggregated to the base station. The collected mobility data logs at the base station contain node movement traces of many participating nodes. In addition to their scientific value, these data logs can be used to extract useful mobility metrics, such as routing lifetime [128]. By feeding such extracted mobility metrics into our model, one can easily predict routing performance and dynamically adjust protocol configurations as necessary. Based on model outputs, proper protocol adjustment decisions are then disseminated to each participating node in the network via a protocol such as one proposed in [71]. This process continues as new mobility data is collected. By introducing such a **feedback**

loop, we create a network that can optimize itself and run well autonomously over long periods of time. Previous approaches (Figure 2.1(a)), however, remain fixed once deployed and cannot change significantly when mobility varies.

2.2 Background and Related Work

2.2.1 Background

MANET routing

A great deal of research has been conducted on multihop routing in traditional MANETs. Direct wireless communication between a sender and a receiver may not always be desirable since it incurs prohibitively high transmission power that grows exponentially with the radio range. Since each node can be both a router and a data source, the use of relay nodes can help to reduce energy consumption. *Multihop routing* is therefore widely used in MANETs in which mobile nodes cooperate to establish network connectivity and conduct routing in the absence of any infrastructure support. It provides wide coverage as well as mobility support by hopping over multiple ad hoc wireless links.

Based on their methods of route discovery and maintenance, the protocols developed in this research can be divided into three classes: reactive protocols, proactive protocols and hybrid ones. Reactive protocols are suitable for mobile networks because routing is conducted on demand. Among reactive protocols, Dynamic Source Routing (DSR) [56] and Ad hoc On-Demand Distance Vector routing (AODV) [91] are the most extensively studied. These protocols have been successfully implemented, and widely tested in different scenarios, such as mesh networks and wireless hot-spots. Their concepts have also been adopted in many commercial products.

DSR

DSR is a reactive source routing protocol, which consists of two major components: *route discovery* and *route maintenance*. Both components operate entirely on-demand. During route discovery, a node scouts through the network to find routes to an intended destination. Route maintenance is the process by which the sending node determines if the route used is broken and takes recovery actions when necessary.

When a data packet arrives, a request is made to the route cache for its intended destination. If a route is found (a cache hit), the packet is forwarded to the next node along the route. We call the selected route the *candidate* route. All other routes with the same destination are *auxiliary* routes. If the request misses (a cache miss), a route discovery is initiated, with the packet forwarded along the newly discovered route if found. If the request has a hit but the candidate route is stale (a “false hit”), delays are introduced at intermediate hops to fix the broken route. Even worse, the packet may be dropped if the error cannot be fixed en route.

However, due to their reactive attributes, for networks with sparse connectivity, DSR may perform poorly when node mobility becomes highly varying.

2.2.2 Related Work

There is a large body of literature on combating mobility challenges in mobile networks and we only discuss the most relevant work here.

Mobility Models

In terms of prior work with analytic techniques, the bulk of modeling has concentrated on the analysis of MAC protocols, for either single-hop [18] or multi-hop networks [19].

These works provide solid understanding of behavior in the MAC layer. However, overall routing behavior cannot be explained without reference to higher layer protocols. A model beyond the MAC layer is critical for understanding end-to-end routing behavior. Several related analytical models in higher networking layers have been proposed. Zhou *et al.*[148] developed performance models of reactive routing in the network layer for an unreliable static sensor network. Their main analytical results are with regard to control overhead, while ours are with regard to overall performance metrics. Viennot *et al.* [124] also proposed a model for analyzing protocol control overhead, but as the aim is to be general, many important details are missing and it becomes difficult to isolate the impact of node mobility, not to mention leveraging the model for protocol optimizations.

Research by Shah *et al.* [107, 108] has modeled data delivery rates in a mobile sensor network. Their model uses an asynchronous store-and-forward communication pattern suitable for Delay Tolerant Networks [30]. Therefore, no end-to-end route semantics are considered. Samar *et al.* [104] develop an analytical framework to investigate the timing behavior of the communication links, while our study is based on route lifetime. They evaluate their framework in a synthetic random environment, while we use realistic mobility traces.

Mobility Characterization

Recently, many mobility datasets from real-world applications have been collected and archived at CRAWDAD (the Community Resource for Archiving Wireless Data At Dartmouth) [65] and are publicly available to the research community. These data provide opportunities for deeper understanding of real-world mobility characteristics. They can also be used to derive mobility models that can be integrated into various simulation tools for protocol evaluation. Many of the datasets confirm that a simple random mobility model is far from ideal in characterizing real-world mobility properties [62, 142]. Based on this observation, we use the mobility model from the ZebraNet project for our evaluation, which is also archived at CRAWDAD [129].

Work by Bai *et al.* [5] studies various mobility statistics in a mobile network, while we focus on mobility metrics that have a direct impact on routing performance. Follow-up work by the same authors provides a detailed study of how mobility impacts path duration statistics in MANETs [101]. None of the above works, however, tie mobility to protocol behavior as ours does.

2.3 Model Overview

In this section, we present an overview of the route cache model in the context of DSR routing. A more detailed mathematical construction is presented in Section 2.4. Figure 2.2 sketches all quantities of interest and their relationships.

2.3.1 Model Outputs

Our route cache model essentially outputs three steady-state probabilities, the probability of cache hit (π_h), miss (π_m), and false hit (π_f). Since route cache access is on the critical path of routing, its access behavior is tightly connected to and reflective of the overall routing performance. This is illustrated in the upper dashed frame in Figure 2.2. In this study, we consider two performance metrics: *packet delivery rate*, defined as the fraction of successfully delivered data packets and *average data latency*, defined as data latency averaged among all data packets delivered.

Intuitively, increasing π_h improves both packet delivery rate and data latency. The higher the π_h, the better the two performance metrics since packets are delivered along

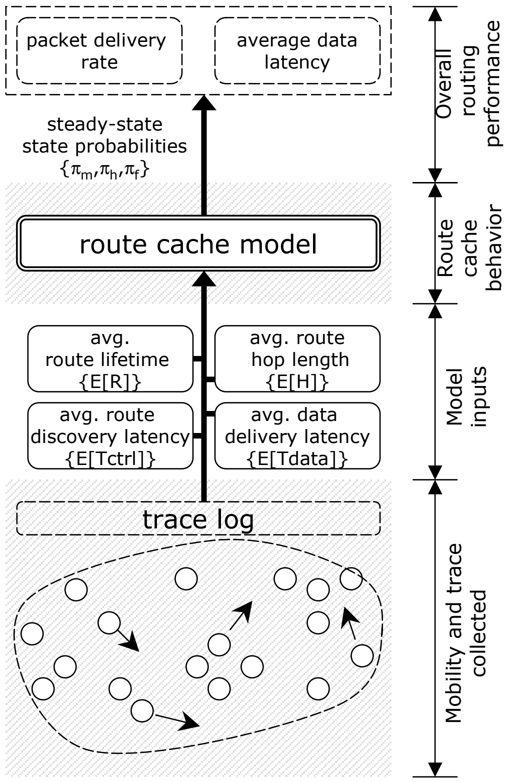

packet delivery rate
average data latency
steady-state state probabilities {π_m,π_h,π_f}
route cache model
avg. route lifetime {E[R]}
avg. route hop length {E[H]}
avg. route discovery latency {E[Tctrl]}
avg. data delivery latency {E[Tdata]}
trace log
Overall routing performance
Route cache behavior
Model inputs
Mobility and trace collected

Figure 2.2: Modeling framework and data flow.

good routes and do not need to wait for re-discoveries and error recoveries. Increasing π_m, on the contrary, has a degrading effect on data latency because a route discovery has to be followed that adds to the total latency. For packet delivery rate, π_m's impact depends on the current traffic conditions and the queueing behavior of the protocol. If no packets are dropped due to these factors, the packet can be successfully delivered after a route discovery with valid replies. π_f, however, always has a negative impact on routing performance because following a stale route will always incur more processing overhead to repair such errors. Very likely, packets will be dropped by failing to fix such errors en route. π_f can happen during packet forwarding, route reply, and packet salvaging by providing a stale route, causing poor packet delivery rate and increased data latency.

Table 2.1: Model input.

Notation	Description
$E[R]$	Average route lifetime
$E[T_{\text{ctrl}}]$	Average route discovery latency
$E[T_{\text{data}}]$	Average data delivery latency
$E[L]$	Average route hop length

However, collecting π_f in practice is hard, if not impossible, after a protocol is deployed. Since our model exposes the probability of being in the *false hit* state, it can be used as an indicator of such *non-sensical* protocol behaviors. By factoring such knowledge into a system, these otherwise unachievable metrics can be leveraged at system runtime.

2.3.2 Model Inputs

Table 2.1 lists the input parameters for our route cache model. The parameter $E[R]$ denotes the average lifetime of routes in a network and its inverse denotes the rate at which valid routes become stale. The shorter the average route lifetime, the more frequently a route breaks. Since a route breakage triggers route maintenance in a reactive protocol[1], $E[R]$ is used to characterize node mobility. Previous work [128] has shown that route lifetime is useful in capturing mobility properties. We choose to use an average here for its simplicity and amenability to analysis. The use of route lifetime *distributions* will improve model accuracy, as further discussed in Section 2.4.3.

The parameters $E[T_{\text{ctrl}}]$ and $E[T_{\text{data}}]$ are used to capture the timing behavior of two critical protocol-related operations. $E[T_{\text{ctrl}}]$ denotes the average latency of a broadcast-based route discovery process and $E[T_{\text{data}}]$ denotes the average data packet delivery la-

[1]There are other factors, such as having multiple paths in the route cache, that influence the triggering rate of route maintenance operations. They are further discussed in Section 2.4.

tency. The route discovery process populates the route cache and provides routes for data packets. Because data packets are used implicitly for signaling routing errors in DSR, the data delivery latency determines how quickly a stale route in cache is detected.

The parameter $E[L]$ denotes the average route length in terms of hop count. It depends on node mobility and traffic pattern and is useful in determining the delay for detecting a route error, as discussed in Section 2.4.2.

We estimate $E[R]$ by bookkeeping route creation and dead events in the route cache. We only track routes that have existed at least once in the cache for the following reasons: First, only routes stored in the cache can influence the route cache behavior. Second, naively bookkeeping all potential routes is computationally expensive. For a network with n nodes, the number of potential routes is on the order of $n!$, which grows exponentially as n becomes larger.

$E[T_{\rm data}]$ can be measured by timestamping data packet departure and arrival events. $E[T_{\rm ctrl}]$ can be collected in a similar way as $E[T_{\rm data}]$. However, since it measures the latency from when a route request is sent out until a **valid** reply is received in our model, we need to check the validity of discovered routes. This is possible with off-line trace processing where omniscient knowledge of route validity is available. $E[L]$ can be measured by recording the number of hops traversed for each packet successfully delivered.

2.4 Route Cache Model

In this section, we describe the construction of our route cache model. Model parameters are listed in Table 2.2.

Table 2.2: Model parameters.

Notation	Description
κ	Rate of a route becoming stale. Also the transition rate out of state *H*.
μ_1	Stale route detection and invalidation rate. Also the transition rate out of state *F*.
μ_2	Route recovery service rate. Also the transition rate out of state *M*.
p_{xy}	The transition probability from state x to state y. Both x and y can be one of the following: h, m, or f.

2.4.1 Assumptions

A1. (Cold start miss) We assume no cold start misses once the route cache reaches steady state.

A2. (Capacity miss) We assume no capacity misses. This is reasonable because capacity misses are independent of mobility-induced misses and can be eliminated easily by increasing cache size.

A3. (Channel models) A noisy channel may reduce the actual route lifetime due to transmission failures. We do not consider route breakage related to this factor and leave it as a future research direction.

A4. (Traffic pattern) We only consider saturated traffic workload with all nodes continuously pumping data to the base station. This assumption matches a large range of real-world traffic patterns, such as the one used in ZebraNet. For on-demand protocols whose operations depend on traffic distribution, the correlation between route cache behavior and traffic is lowered.

A1-3 allow us to assume that the average route breakage rate κ depends only on node mobility, which is abstracted as route lifetime timers. We will show later in this section that even with such simplifications, our model still produces accurate results.

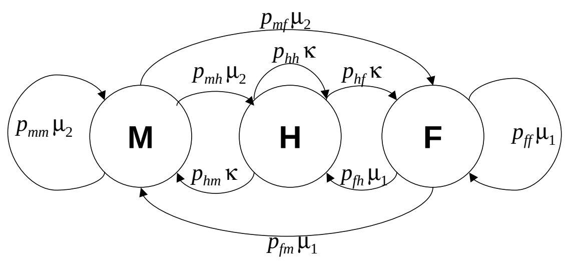


Figure 2.3: The Discrete Time Markov Chain model for a single node.

2.4.2 Model Mechanics

Figure 2.3 illustrates our three-state Discrete Time Markov Chain model for a route cache. In state **M**, the node has no candidate path for an initiating packet and a cache miss occurs. In state **H**, the node has a valid candidate path for an initiating packet and a normal hit occurs. In state **F**, the node has a candidate route that is stale due to mobility and a false hit occurs.

Virtual Detection State (F)

One major contribution of our model is that we add a state F, the false hit state, which does not exist in a realistic protocol. Thus, we name this state the *virtual* detection state. For many reactive ad hoc routing protocols, including DSR, it is impossible to detect route failures instantaneously and they will inevitably enter this artificial state. Having such a virtual detection state expose valuable information that is essential to routing performance. Such information is not possible with traditional approaches.

Rate of Cache Staleness (κ)

The rate of cache staleness, or the rate transiting out of state H, is κ. Intuitively, for some period proportional to $E[R]$, the current candidate route will break which leads the node to state *F* or *M*, depending on factors such as the route discovery and maintenance mechanism used, node mobility and traffic workload. The average actual route lifetime should be smaller than $E[R]$ because it only represents the average lifetime in the route cache. Therefore, we need some adjustment to calculate κ using $E[R]$. In our model, we estimate κ as $\frac{1}{\gamma E[R] - \frac{1}{\mu 1}}$ with γ a constant denoting the ratio between the actual lifetime to $E[R]$. We use a γ of 0.5 in our study, assuming that when a route is selected for routing, its residual lifetime is uniformly distributed between $(0, E[R])$. We subtract $1/\mu_1$ from $E[R]$ because during route recovery, the elapsed route lifetime cannot be used for routing.

Route Discovery (μ_2)

The average route discovery rate is denoted as μ_2. It is also the transition rate out of state M. We estimate μ_2 simply as $\frac{1}{E[T_{\mathrm{ctrl}}]}$.

Stale Route Detection and Recovery (μ_1)

With the introduced virtual detection state F, we can model the rate that a node detects an invalid route, which is also the transition rate out of state F. We denote this as μ_1.

Route error detection in a reactive routing protocol is divided into two phases: a *negative detection phase* and an *active error notification phase*. In the negative detection phase, data traffic is used implicitly to detect a link error when the packet reaches the broken link. In the active error notification phase, a control packet is sent to notify the source of this error.

Therefore, the negative detection latency $E[T_{\mathrm{negd}}]$ depends on the number of hops to traverse along the broken route until the packet reaches the broken link. If we assume that the probability of route breakage is distributed uniformly over all hops from the source to the destination, we can estimate the average hop count traversed before a route breakage as $\sum_{i=1}^{E[L]} i = \frac{1}{2}(E[L] + 1)$. Therefore, the negative detection latency can be estimated as $E[T_{\text{data}}] \times \frac{E[L]+1}{2E[L]}$. We can calculate the active error notification latency $E[T_{\text{actn}}]$ in a similar way. The only difference is that $E[T_{\text{ctrl}}]$ is a two-way delay that spans $2E[L]$ hops in total. Therefore, $E[T_{\text{actn}}]$ should be calculated as $\frac{1}{2}E[T_{\text{ctrl}}] \times \frac{E[L]+1}{2E[L]}$. These two latencies add up to the average route recovery latency and μ_1 is estimated as $\frac{1}{E[T_{\text{negd}}]+E[T_{\text{actn}}]}$.

Influence of Protocol Designs

In this section, we discuss the influence of protocol designs on state transition probabilities.

In our model a transition out of state M only happens after a route is discovered, so p_{mm} should be 0. For discussion convenience, we directly denote the probability from state M to state H as p_m and the probability from state M to state F as $1 - p_m$. Thus, p_m represents the probability that a route reply is valid and $1 - p_m$ the probability that a route reply is stale. Since we estimate μ_2 using only valid route replies (i.e., only a valid route reply finishes a route request), p_m is 1 in our case.

Since a route will always enter the virtual detection state due to the reactive maintenance mechanism, p_{hh} and p_{hm} are 0 and we denote the probability from state *H* to state *F* as p_h. Thus, $p_h \cdot \kappa$ represents the rate of a broken route not being detected immediately. In our model, p_h is approximated as 1 because DSR mainly depends on data traffic for route error detection.

The parameters p_{fm}, p_{fh} and p_{ff} denote the three transition probabilities out of the false hit state. They all depend on the number of backup routes available when the candidate route breaks because only when there is no route to the destination does the node enter state M. Otherwise, the breakage of the candidate route will not incur a new route request and the state may transit to either H or F, depending on the validity of the new candidate route selected. Therefore, p_{fm} represents the probability of having no backup routes when the route being used is invalidated. Hence, $p_{fm} = 1$ is the case where there is only one route for each destination. Whenever this route is broken, it enters state M. On the other hand, $p_{fm} = 0$ is the case where there are always valid auxiliary routes when the route being used is invalidated.

State Probabilities

In this section, we calculate the equilibrium state probabilities of the model, denoted as $\boldsymbol{\pi} = [\boldsymbol{\pi_m}, \boldsymbol{\pi_h}, \boldsymbol{\pi_f}]$.

We simplify the mathematical calculation by pre-determining parameters that can directly be estimated from the protocol behavior, which are p_h and p_m. From earlier discussions, both of them should be equal to 1 for DSR. Therefore, we have the following global balancing equations for the steady state of our Markov chain model, which can be solved to produce the limiting state probabilities:

$$\left\{ \begin{array}{rcl} \pi_h \kappa & = & \pi_f (1 - p_{ff}) \mu_1 \\ \pi_f p_{fm} \mu_1 & = & \pi_m \mu_2 \\ \pi_m + \pi_h + \pi_f & = & 1 \end{array} \right.$$

2.4.3 Route Cache Model Validation

In this section, we present model validation results against *ns-2* simulations using the Random Waypoint (RWP) model. Our approach can also be used with other mobility models since only high-level mobility metrics are used in our model. In Section 2.5, we will validate our model on a real-world mobility. We seek to study (i) how close

Table 2.3: Route cache model validation.

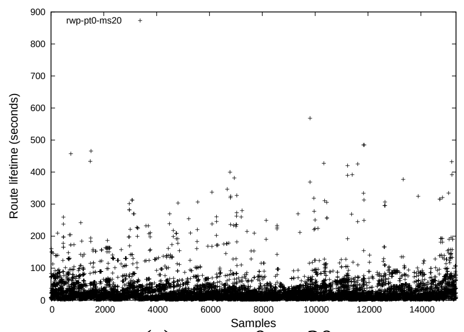
Scenario	π_m			π_h			π_f			Running Time		
	Sim	Model	%diff	Sim	Model	%diff	Sim	Model	%diff	Sim	Model	sim/model
rwp-pt0-ms20	0.124	0.136	9.6%	0.526	0.538	2.3%	0.349	0.326	6.6%	22:14m	49s	27
rwp-pt10-ms20	0.243	0.238	2.1%	0.354	0.358	1.1%	0.403	0.404	0.2%	13:22m	54s	15
rwp-pt20-ms20	0.166	0.154	7.2%	0.456	0.475	4.2%	0.378	0.371	1.9%	24:37m	57s	26
rwp-pt20-ms1	0.075	0.053	29.3%	0.884	0.906	2.5%	0.042	0.41	2.4%	17:47m	39s	27

our analytical model results are to simulated outcomes of DSR, under different mobility scenarios, and (ii) how our model parameters affect its accuracy.

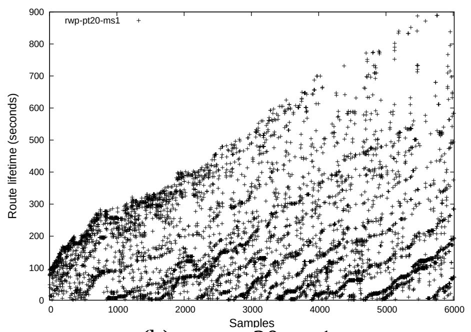
We simulate a network of 50 nodes in a 1100m$\times$1100m grid. Each node has a radio range of 250m. Initially, nodes are randomly distributed across the defined area. We generate 30 communication pairs randomly and use a packet rate fixed at 2pkt/s. We do not adopt a higher injection rate because we need a network sufficiently provisioned such that the effects of mobility are isolated from effects of congestion [46]. We use UDP traffic in packets of 512 bytes. Traffic is injected from the 900s mark to populate the route cache and all metrics are measured starting at the 1000s mark. The first 900s is used for the mobility model to reach its steady state, a method proposed in [141] to fix the deficiency of RWP not having a steady state. Other simulation setups, including the radio propagation model, the MAC protocol used, and link bandwidth are the same as those used in [12].

Validation Results

Table 2.3 compares the analytical results against simulation results for four mobility scenarios. Each scenario is represented as rwp-pt$x$-ms$y$ with x its pause time and y its maximum speed. For a moderately- to highly-mobile network, our model is reasonably accurate with error rates less than 10%. This indicates that in steady-state, our model successfully captures the state of the route cache in a mobile environment. For static scenarios such as rwp-pt20-ms1, however, our model has a much higher error rate.



(a) rwp-pt0-ms20



(b) rwp-pt20-ms1

Figure 2.4: Route lifetime distributions for different mobility scenarios.

This high error rate can be explained by looking at the model inputs. Figure 2.4 illustrates the distribution of route lifetimes observed by a randomly selected node (14) for both rwp-pt0-ms20 and rwp-pt20-ms1. The observations for other nodes are similar. The samples are collected by post-processing the simulation trace. For a moderately-mobile to highly-mobile network, route lifetimes tend to be in the same order of magnitude, as shown in Figure 2.4(a). For a less mobile network, however, route lifetimes have very high variability, as shown in Figure 2.4(b)[2]. In such a network, skew will be introduced by representing route lifetimes as an average. This problem can be solved by using separate estimations for short and long lifetimes. More generally, one could include distributions of route lifetimes in the model.

Table 2.3 also illustrates the savings in running time to derive the same quantities of interest. Model computation time is negligible, while input gathering time dominates. It should be noted that when the model is dynamically deployed, input gathering time is spent on running scripts on trace files, which normally takes less than 1 minute. Therefore, the total elapsed time using our model is less than 1 minute. On the contrary, it

[2]While there appears to be a correlation between simulation time and route lifetimes, this is an artifact of our statistics gathering method: routes that last longer than the simulation time cannot be tracked.

usually takes more than 15 minutes to finish one simulation run using traditional approaches.

Future Refinements of the Model

Overall, the model's accuracy is already quite good. Nonetheless, there is still room for improvement. The modest disagreements between analytical and simulated results can be explained by certain simplifying assumptions regarding state transition probabilities. We discuss these below.

The first source of error has to do with the transition probability from state *H* to *M*. For the implementation of DSR in *ns-2*, this probability is greater than 0, due to various protocol optimizations not considered in the model. One such optimization is route error propagation, which spreads route error messages aggressively to suppress their propagation. Thus, a route recovery process can be finished without incurring the two-phase operation. Another optimization is cache purging that times out a route after some duration. This also may lead a node in state *H* directly to *M*.

The second source of error arises in the presence of a false route reply. In our model, we only account for valid route replies when calculating $E[T_{\text{ctrl}}]$. This approach may under-estimate the probability entering state *F*. In other words, we assume there is no state transition from *M* to *F*. In contrast, such effects are present in our simulations. This explains why for most scenarios, π_f calculated by our model is smaller than that from simulation.

2.5 Case Study

In this section, we validate our model using real-world mobility based on the wildlife tracking application we mentioned previously.

2.5.1 Validation Using Real-World Mobility Data

Our mobility trace is collected from a mobile sensor network deployed in January 2004 by the ZebraNet group [129]. A number of collars (sensor nodes) are attached to zebras. Each collar recorded its GPS data every 8 minutes for a total of 32 hours. Due to extreme weather and waterproofing issues, as well as antenna problems, only one tracking collar returned uninterrupted movement data for the whole 32-hour duration. Due to such limitations, we extended the collected data to create a semi-synthetic mobility model as follows. We collect node speed and turn angle *distributions* from the observed data. Then we create other node movements by uniformly selecting from the node speed and turn angle distribution collected in the first step. Next, we cast the trace data into a RWP model that can fit into the *ns-2* simulator. Although this approach may miss some temporal correlation information between zebras, it is one step closer to reality.[3]

Originally, the nodes move in an area of 6km$\times$6km. We scale the area size to 1km$\times$1km and randomly distribute the nodes in the defined area. In order to calculate metrics like cache false hit rate, we also incorporate other needed information about connectivity and shortest route length at any instance between all communicating pairs, so that the trace file can be directly used in *ns-2* simulations. The rest of the simulation configuration is the same as that described in the previous section.

[3]From extended data collection in a second, June 2005 deployment, we found that there is little node correlation in movements, and thus our assumption here is valid.

Table 2.4: Validation results for a real-world mobility.

Category	Sim	Model	%diff
π_m	0.130	0.133	2.3%
π_h	0.509	0.486	4.7%
π_f	0.361	0.376	4.0%

Table 2.4 shows that our three model outputs have error rates below 5%, validated against ns-2 simulation. This indicates that our model can achieve good accuracy in capturing route cache access behavior even for real-world mobility.

2.5.2 A Case for a Model-Driven Dynamic Protocol

In this section, we present a case study demonstrating how to leverage our model to drive adaptive routing decisions on-the-fly. Although this example is based on DSR, our model also works for other route-cache based protocols such as Directed Diffusion (DD) because all details discussed so far are also applicable to DD [51].

DSR uses route caching extensively in both route discovery and route reply. It adopts a passive route maintenance mechanism for fixing stale routes. The problem with such a scheme is its slow response to mobility changes. Given that all routing decisions are based on route cache state, the performance may suffer from using stale information. By exposing the route cache states, our model helps to predict route caching performance in a timely fashion and guide protocol adjustment when necessary.

Specifically, we show how route cache reply options can be switched on and off dynamically to improve routing performance by leveraging π_f. This idea can be used for other optimizations, such as route discovery backoff, given proper models for those components. The mobility used is derived from the zebra trace with node speeds varied. We divide the mobility trace into three phases. Phase 1 is from 1000s to 1300s, phase 2 is

Table 2.5: Configuration options studied.

	1st phase	2nd phase	3rd phase
DSR (always-on)	on	on	on
Strategy 1 (off-low)	off	on	on
Strategy 2 (always-off)	off	off	off
Strategy 3 (off-high)	on	off	off

Table 2.6: Data latency comparison.

Traffic rates (Packets per second: pps)	DSR	Strategy 3	%improvement
2	6.2s	5.2s	16%
4	4.1s	3.0s	27%
8	2.7s	2.1s	22%

from 1300s to 1600s, and phase 3 is from 1600s to 1900s. Traffic starts at 900s to populate the route cache. We reduce the node speed to 0.1 of the original speed for phase 1, increase by 3x for phase 2, and increase by 6x for phase 3. The trace produced this way demonstrates a significant variation from one phase to another and is fairly realistic as zebras normally move in walk-run-walk phases [58]. Moreover, we expect such phases of varying mobility to typify many other mobile network scenarios as well.

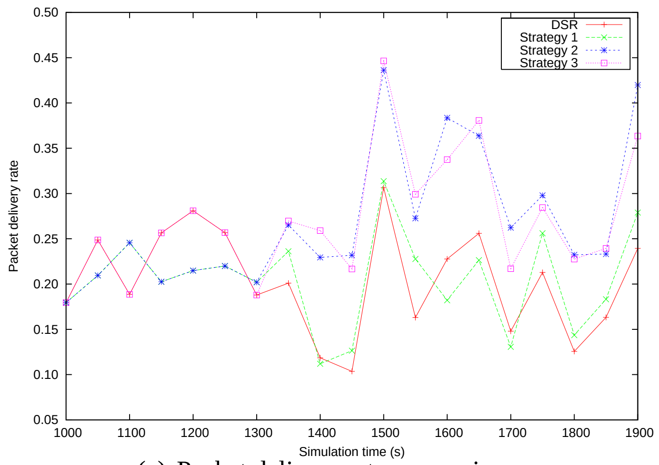
The set of experiments we performed uses a similar configuration as described in the last section, with a total of 50 nodes and 30 constant bit rate (CBR) flows. We use a radio range of 150m here because a 250m radio range for this mobility trace results in severe radio interference in our simulation. We study the instantaneous packet delivery rate and normalized routing overhead for three configurations listed in Table 2.5 and compare their performance with the original DSR. By *instantaneous*, we mean that results shown in the y-axis are not aggregated from the start of the simulation. They demonstrate instant behavior for that period. This allows for a better observation of the adaptation behavior.

The three configurations differ from DSR in their decisions as to when to switch *route cache reply* on/off for the three phases. Intuitively, for a highly-mobile scenario, route cache replies should be disabled because the information stored in the cache is likely to be invalid; using a route cache for answering route requests can lead to inaccurate routing decisions. For a less mobile scenario, where route cache knowledge is normally accurate, enabling route cache replies will increase locality, reduce latency and save resources. Thus, the decision is based on π_f.

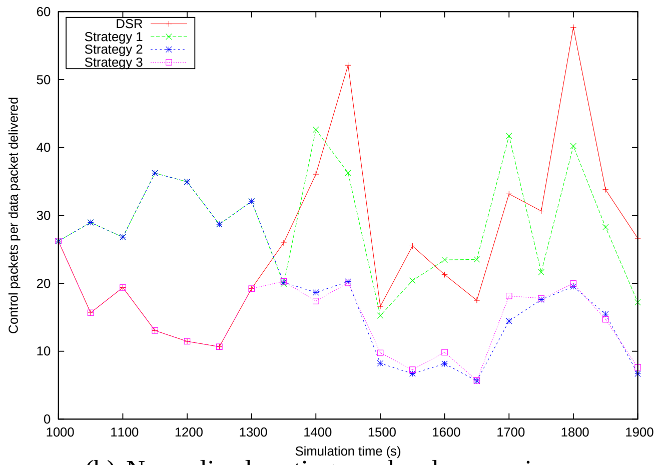
Since phase 1 is very static and phases 2 and 3 are both highly mobile, the strategy that disables route cache replies for phases 2 and 3 and enables route cache replies for phase 1 (Strategy 3 in Table 2.5) should have the best performance, the highest packet delivery rate and the lowest routing overhead and energy consumption. Strategy 1, which has the opposite configuration options to Strategy 3, should have the worst performance. Strategy 2 should stay in the middle because most of the time, it has the right option (for phases 2 and phase 3). The original DSR just switches on route cache replies all the time.

For such an adaptive scheme to work, a node needs to be able to detect the mobility phase changes. For this section, we pre-program such information into the simulation for them to make decisions in a distributed manner. We will discuss a practical phase detection method in Section 2.5.3.

Table 2.6 compares the average data latency between DSR and Strategy 3 using the common set of packets they successfully delivered. Intuitively, Strategy 3 saves route repair time by not following stale routes. However, switching off route cache replies means that it needs more time to get a route because it only accepts replies from the intended destination. On the contrary, DSR saves time by getting a route from other nodes' route cache. If the route obtained is stale, however, it incurs additional delays fixing errors at intermediate hops. Table 2.6 shows that Strategy 3 has a smaller average



(a) Packet delivery rate comparison.

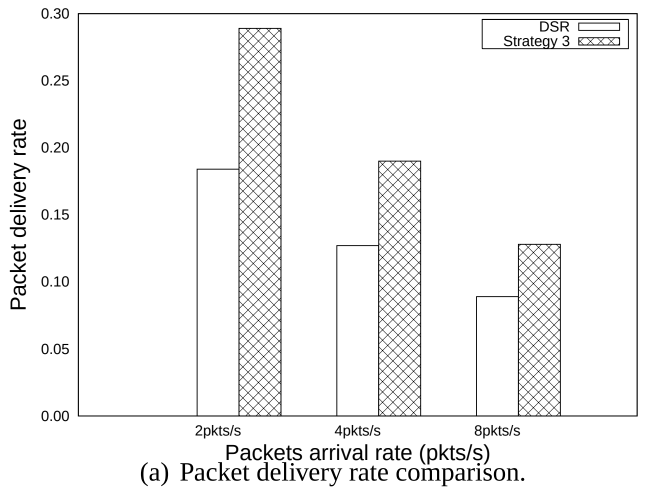


(b) Normalized routing overhead comparison.

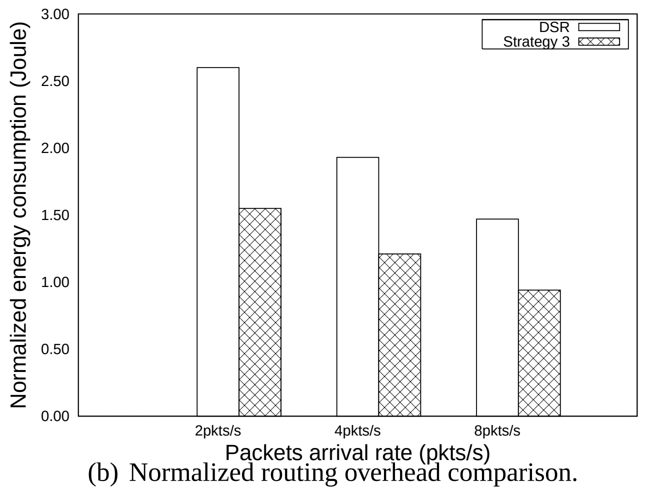
Figure 2.5: Performance comparison of different strategies. Results are collected from simulation and the packet injection rate is 2pps.

data latency for all traffic rates. This indicates that for the mobility trace we studied, it is better to switch off the route cache reply option than to keep it on.

The latency improvement for 4pps and 8pps are both higher than that for 2pps. When more packets are injected into the network, the penalties of using stale routes become higher because the contention for the medium is more severe than at lower packet rates. As a result, the benefits of using dynamic configurations become more salient.



(a) Packet delivery rate comparison.



(b) Normalized routing overhead comparison.

Figure 2.6: Impact of traffic rate.

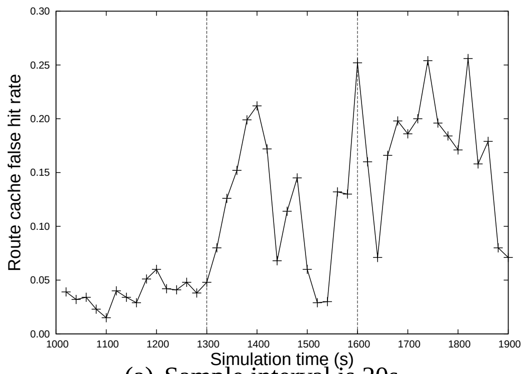
Figure 2.5(a) shows the instantaneous packet delivery rate at an interval of 50s. Figure 2.5(b) shows the routing overhead averaged among all data packets delivered. Routing overhead refers to control packets sent for route discovery and route maintenance. The normalized routing overhead is used as a measure of routing efficiency, including energy efficiency.

For the first 300s from 1000s to 1300s, Strategies 1 and 2 have the lowest packet delivery rates and the highest average overhead. This is because in phase 1 the nodes move very slowly. For such an environment, stale information is very rare and route cache replies should be enabled to maximize locality. For phases 2 and 3, as nodes move quickly, stale information begins to flood the network. In this case, a route cache reply should be avoided because there is a higher probability that the benefits of employing route caching can be overwhelmed by the disadvantages it brings. Strategies 2 and 3, which switch off route cache replies for phases 2 and 3, have a higher packet delivery rates and lower routing overhead than Strategy 1. Because Strategy 3 adapts to varying mobility correctly, it achieves the best of both worlds and has the best performance compared to all other options, including DSR. The improvement in packet delivery rate is consistently higher than 40% and the maximum improvement is up to 120%. The reduction in routing overhead is consistently higher than 40% and the maximum reduction is up to 66%.

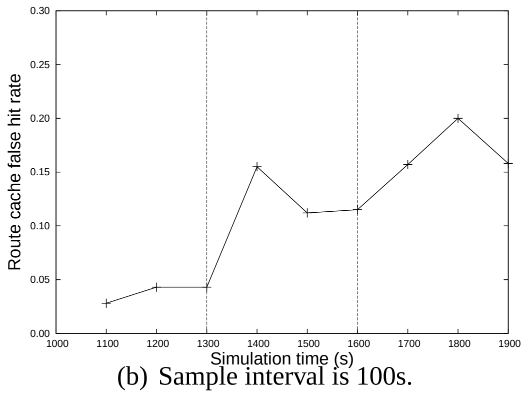
Figure 2.6 illustrates the impact of packet arrival rate on the performance of our dynamic optimization. As the injection rate increases, the demand for bandwidth increases too and we believe that our approach should still outperform a scheme that is unaware of mobility changes. Simulation results shown in Figure 2.6 affirm this.

2.5.3 Detection of Mobility Changes Using π_f

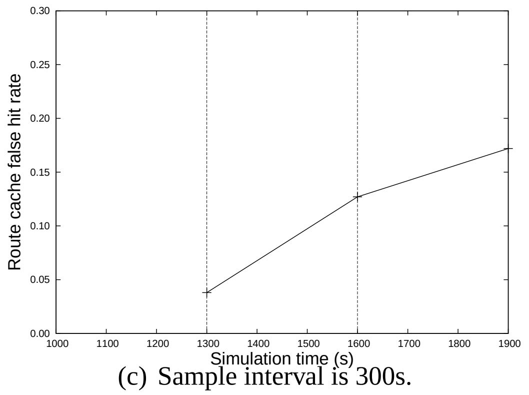
In this section, we propose a practical phase detection method using π_f. It uses a feedback loop as introduced in Section 2.1. When enough mobility data are collected at the sink, we extract mobility metrics as input to the route cache model. Since all input parameters can be obtained by running `awk` and `python` scripts on the collected mobility trace, this process takes only tens of seconds on a modern PC. We then use our model to output π_f, which takes only a couple of seconds at most. This obtained information is disseminated



(a) Sample interval is 20s.



(b) Sample interval is 100s.



(c) Sample interval is 300s.

Figure 2.7: π_f changes for DSR in response to mobility changes.

to all nodes in the network through a data dissemination protocol. A decision is then made at each node regarding its caching strategy.

Figure 2.7 shows instantaneous π_f in our simulation, with different sampling intervals (epochs). Each point represents the π_f re-evaluated at the end of each epoch. Fig-

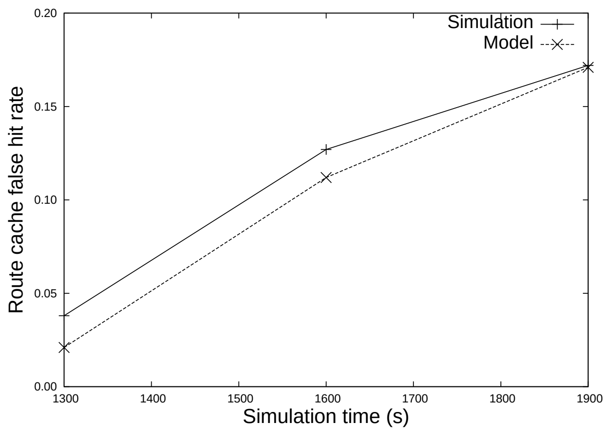


Figure 2.8: Comparison of π_f changes detection (simulation vs. model). The epoch length is 300s and the packet injection rate is 2pps.

ure 2.7(a) shows π_f with a sampling interval of 20s. There are two jumps with the first one starting at 1300s and the second starting at 1600s. They conform to the mobility changes in 1300s and 1600s, respectively and are emphasized using two dotted lines. Because the sampling rate is very high, the variation is pretty high compared to variations using longer sampling intervals. Figure 2.7(b) shows the results for a sampling interval of 100s. The variation is much smaller and the changes in π_f are consistent with changes in mobility. Figure 2.7(c) shows the results for a sampling interval of 300s, which exactly matches the three mobility phases. This in turn indicates that there is a salient change in π_f in response to mobility changes. The results demonstrate that π_f can be used for predicting changes in mobility with reasonable accuracy.

Figure 2.8 compares the estimation of instantaneous π_f using our model to that using simulation for a sampling interval of 300s. As the figure shows, the estimate from our model matches that from simulation very well. This demonstrates that our model can aptly capture the changes in mobility at runtime with high accuracy.

To fully take advantage of such prediction capability, the epoch length needs to be short enough such that the current prediction reflects the mobility in the next epoch. On the other hand, the epoch length needs to be long enough such that the gathered inputs to the model can guarantee accurate model outputs. This study of optimal epoch length is left as future work.

Finally, we compare the running time to derive π_f by model to that by simulation using the semi-synthetic mobility trace as described in Section 2.5.1. The simulation took **13:05 minutes** on a machine with 2.2GHz Pentium 4 processor and 512MB RAM. However, it took only **25 seconds** to output π_f using our model. Simulation time becomes even longer when dynamically trading off between different parameter configurations because several simulation runs are then necessary.

2.5.4 Discussion

While our model and its use already demonstrate significant performance improvements, there is still room for future refinements. We discuss some of them here.

First, our current approach requires the base station to collect node mobility traces for analysis. This is a challenging task for a MSN, even a modest-size one. Second, our current model only derives steady-state probabilities, which requires a certain mobility phase to be long enough to be observed. We do not view this as a significant weakness, since short-lived mobility changes are not likely to be worth optimizing for. Third, our model tries to use a single metric (the false hit) to capture the route cache behavior for nodes distributed across the network, which works for a homogeneous network wherein nodes experience similar mobility patterns. However, for other realistic mobility patterns, a distributed algorithm may perform better.

Although our approach has such limitations, it offers new opportunities for using analytical models in real-world setting and further work is anticipated to improve on this.

2.6 Conclusions

In this chapter, we presented an analytical model of route cache for DSR-like reactive protocols. We illustrated how to use the model to expose crucial situation information (mobility phase changes in this case) and drive dynamic protocol to adapt to varying mobility, using a real-world mobility. When validated against detailed network simulations, our model produces fairly accurate results with typical errors less than 5% for a real-world mobility and less than 10% for synthetic RWP-based mobilities. To the best of our knowledge, our work is the first to model the behavior of a route cache for MANETs and mobile sensor networks. Our model-driven adaptation can improve instantaneous packet delivery rate by up to 120% and data latency by 16-27%.

Chapter 3

A Supervised Learning Approach to Routing Optimizations

Routing protocols in sensor networks maintain information on neighbor states and potentially many other factors in order to make informed decisions. Challenges arise both in (a) performing accurate and adaptive information discovery and (b) processing/analyzing the gathered data to extract useful features and correlations. To address such challenges, this chapter explores using supervised learning techniques to make informed decisions in the context of wireless sensor networks. We investigate the design space of both offline learning and online learning and use link quality estimation as a case study to evaluate their effectiveness.

3.1 Problem and Solution Overview

Many critical applications in wireless sensor networks fundamentally rely on fast, efficient, and reliable data delivery. In order to overcome the inherent unreliability of sensor network communication links, communication protocols increasingly employ intricate and situation-aware adaptations to identify good routes and to determine resource-efficient methods for handling data.

The difficulties in situation-aware network adaptations are two-fold. First, some adaptation techniques are hard-wired heuristics based on observations of a few stylized types of network problems and their solutions. The more problems one envisions, the more complicated the protocol becomes in trying to adapt to them. Second, environmental factors interact in such complex ways that it can be difficult to identify correlations and crisply define the problem scenarios to protect against. Based on these observations, we explore using machine learning techniques to improve situation-awareness in order to optimize sensor network communication.

Machine learning is an effective and practical technique for discovering relations and extracting knowledge in cases where the mathematical model of the problem may be too expensive to derive, or not available at all. Supervised learning is a particular case when the inputs and outputs are both given. For example, inputs might include node-level and network-level metrics, such as buffer occupancies, channel load assessments, and packet received signal strength. Output may be the expected number of transmissions over the link where the packet is received. Essentially, we aim to use machine learning to *automatically* discover correlations between readily-available features and the quantity of interest. Supervised learning is an effective learning technique in solving this type of problem.

We manage the resource constraints of sensor networks by employing machine learning in a two-phase method: an offline training phase followed by an online classification. Offloading the training task from the sensor node reduces the processing, communication, and energy requirements of the node. The resulting classifiers to be used online are both strikingly lightweight and strikingly effective. For the case studies we have examined, our supervised learning techniques result in prediction accuracies of 80% or more, with false positive rates between 4.1% and 11.3%, and with essentially no compute overhead during their online phase.

We evaluate the effectiveness of our approach using link quality estimation as a case study. For this purpose, we present MetricMap, a data collection protocol atop MintRoute that predicts link quality using knowledge learned from the training phase when the network is highly congested. Evaluation of a prototype implementation in TinyOS on a real-world sensor network shows that MetricMap can improve packet delivery rate and fairness over existing approaches by up to a factor of three under moderate to heavy traffic load. The compactness of our classifier makes it suitable for resource-constrained situations.

For a network with highly varying link qualities, incorporating such new pieces of information is of critical importance to the success of the learning task. For this purpose, we investigate the possibility of using online learning to efficiently maintain a high-accuracy classifier. The attractive property of online learners is that they do not need to process the entire data set at the same time, but can work incrementally with new data coming in. This is more resource- and computation-efficient than traditional batch learners. Our results show that the online learner we used achieves an accuracy similar to traditional batch learners for a link quality data set collected from a real-world sensor network testbed.

3.2 Background

3.2.1 Link Quality Estimation

Wireless sensor networks are very different from wired networks in that the link quality fluctuates greatly as a consequence of interference and propagation dynamics. Therefore, developing efficient routing in sensor networks requires the establishment of high quality paths, which in turn entails accurate knowledge of link quality. In this section, we briefly review the mechanisms behind existing link quality estimation methods, including both software-based and hardware-based ones. We also explain how they fail to function when the traffic rate becomes high. This motivates our work on new approaches based on machine learning.

Software-based Estimation

A few software-based link metrics have been proposed in the past. Route metrics are built atop link metrics to capture end-to-end forwardness. For example, ETX [27], also proposed in MintRoute [136], is one such route metric. It is defined as the expected number of transmissions (including retransmissions) for a successful end-to-end data forwarding and hop-by-hop acknowledgment.

We focus here on the snooping-based method adopted by MintRoute[1]. It defines link quality as

$$etx(l) = \frac{1}{p_f(l) \times p_r(l)}$$

with $p_f(l)$ the forward probability of link l and $p_r(l)$ its reverse probability. $p_f(l)$ is calculated using the ratio of the number of data packets received to the total number of

[1]The difference between the two approaches is studied in [143].

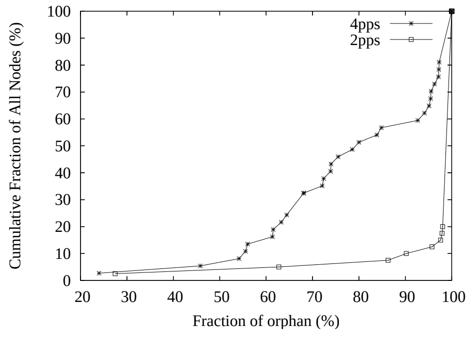
data packets transmitted over l. $p_r(l)$ is calculated as $p_f(\bar{l})$ with $\bar{l}$ the reverse link of l. The route metric of a n-hop path p is then calculated as $ETX(p) = \sum_{i=1}^{n} etx(l_i)$, the total expected number of (re)transmissions along the path.

However, in many high data rate applications [67, 88], snooping-based link quality estimation works very poorly, as we will quantify shortly. For example, consider the high data rate structure monitoring application discussed in [88]. Due to structural vibration damping effects, a very high data sampling rate is required, which is estimated to be at least 200Hz. Therefore, the data rate can be as high as 9.6Kbps per node with each node sampling 16-bit in three spatial dimensions. Even with in-network processing techniques, such as data aggregation [41, 74], compression [102] and coding [92, 96], the expected data rate is still very challenging for current systems to cope with.

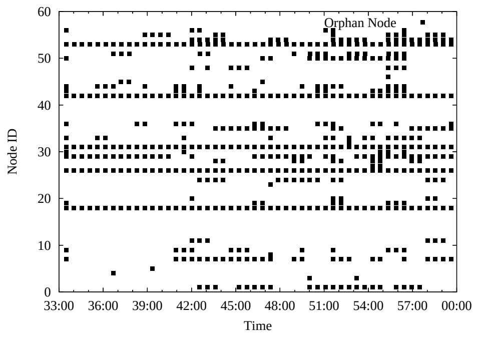
To demonstrate the impact of high traffic on ETX's link quality estimator, we evaluate the performance of MintRoute by running the Surge application on MistLab [80], an indoor sensor network testbed of 60 Mica2 nodes. Surge[2] is a data collection application in which each node generates data traffic at a constant rate and sends to the sink via multi-hop routing. We use MintRoute to build the multi-hop data collection tree that chooses a parent (next-hop in the collection tree) based on additive link/path quality estimation. We define orphan nodes as those that have no parent in the collection tree. Figure 3.1 shows that packet delivery rate degrades once the offered load is 2 packets/second (pps) or higher.

Figure 3.1(a) shows the network-wide fraction of orphan nodes with traffic loads of 2pps and 4pps. The percentage of orphan nodes increases quickly with increases in offered load. For a 4pps offered load, 90% of the nodes do not have a parent 50% of the time. This dramatic increase in the percent of orphan nodes is a direct cause of packet

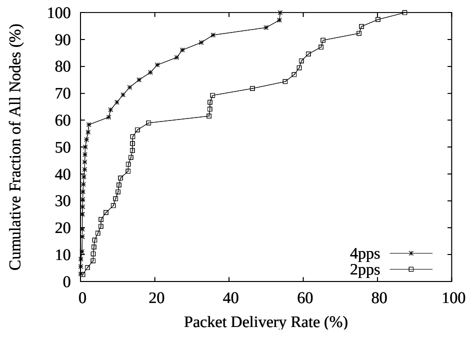
[2]The reference implementation is in the TinyOS CVS repository: `tinyos-1.x/apps/Surge/`.



(a) CDF of percent of orphan nodes.



(b) Spatial distribution of orphan nodes as a function of time.



(c) CDF of packet delivery rate (PDR).

Figure 3.1: Experiment results on a testbed of 60 motes. Orphan is defined as a node that has no parent in the collection tree. The percent of orphans is defined as the ratio of the orphan period to the whole running time. We periodically probe the routing state of a node and estimate this fraction as the ratio of the number of times that the node is an orphan to the total number of probes issued. The first two figures show the spatial and temporal distribution of orphan nodes for different offered load. The fraction of orphan nodes is very high when the offered load is above 2pps, which leads to a lack of routing information and a need for prediction.

losses in the network, shown in Figure 3.1(c). Given a percentage of packets p received from a given node at the sink, the Cumulative Distribution Function (CDF) plots the fraction of sensors that deliver at most p percent of their data to the sink. For the 4pps case, about 60% of all nodes have less than 10% data delivery rate. Figure 3.1(b) plots the distribution of orphan nodes in the network as a function of time. The x-axis shows the experiment timeline in the granularity of seconds. The y-axis is the node ID in the network. Each square dot at (x, y) indicates that at instant x, node y has no parent. In a network with a partitioned collection tree, many packets are transmitted from the edge towards the sink, only to be dropped before they reach their sink.

An examination of the etxs of all nodes shows that a large proportion of links have quality values indicating that very few transmissions can be carried through. As a result, routing is interrupted due to a lack of link quality information. This is directly related to how snooping-based estimation methods behave in an overloaded network. However, since not all links are overloaded, routing can be resumed once an accurate estimation of link quality is in place. We wish to develop link quality estimators that are more resilient in high-traffic settings, and machine learning offers us an efficient way to discover them.

Hardware-based Estimation

The link quality indication (LQI) metric is a characterization of the strength and/or quality of a link over which a packet is successfully received. LQI was introduced in the 802.15.4 standard [2] and is provided by the CC2420, a radio used in many mote platforms, including the MicaZ and Telos motes. It is an integer ranging from 0x00 to 0xff, with the minimum and maximum LQI values associated with the lowest and highest quality signals detectable by the receiver (between -100dBm and 0dBm) . It is reported in [95] that the average LQI closely maps the average success rate of packet transmissions across several links. In this chapter, we use LQI to label link quality in each input/output sample.

3.2.2 Supervised Learning Overview

The goal of supervised learning is to predict the value of an outcome metric based on a number of input metrics [81]. The outcome metric could be numerical or categorical. Learning is performed on a set of training samples. Each sample $\langle x_i, y_i \rangle$ consists of a feature vector x_i and a corresponding class label or numerical value y_i. The feature vector contains measurable features of the system under consideration. If the outcome is categorical, the learning becomes a classification problem. Training a classifier usually involves finding a mapping from feature vectors to output labels so that the overall classification error is minimized on the training samples. A good learner should accurately predict new samples not in the training set. Therefore, given a classification problem, we need to decide (a) what features to measure and (b) what learning algorithm to use to maximize the learning accuracy.

In this chapter, we evaluated two classification algorithms: *decision tree learners* and *rule learners*. We have also tested other classification algorithms, including support vector machines, Bayesian networks, and ensemble methods. Any such learner can be used to train a classifier in our case. However, our results show that decision tree learners and rule learners produce remarkably good accuracy for our case study and they often achieve the highest accuracy among all algorithms studied. Also, due to the complexity and resource concerns specific to sensor networks, we focus on these two learners in the following discussion. A detailed evaluation of them is presented in Section 3.3.

Decision Tree Learners

Decision tree learners are widely used in solving classification problems with classifiers represented as trees. They take a “divide-and-conquer” approach and recursively divide attributes at each internal node in the tree based on information they possess. Leaf nodes represent classification decisions. Pruning methods are used to prevent over-fitting of training data. Although decision tree learners are not always the most competitive learners in terms of accuracy, they are computationally efficient and the results produced can be easily converted to human-readable formats.

Rule Learners

Rule learners are used for learning IF-THEN rules. Like decision tree learners, rule learners work on training samples with similar input/output pairs. However, since the rule-sets learned are disjoint to each other, they usually produce far fewer rules than decision tree learners on the same training set, and have a comparable accuracy. This makes it preferable in scenarios where classifiers need to be used at runtime.

Practical Concerns

Due to the resource constraints in wireless sensor networks, we need to also consider learning efficiency and overhead, in addition to learning accuracy. In particular, we consider the following two factors:

1. *Overhead*: Overhead includes the processing time, energy usage, and the memory footprint, etc. Since in our proposed method, training is conducted offline, usually on a PC or server, computing, energy and memory overhead in the training process should not pose a problem. The overhead of conducting online classification and

feature collection, however, is our major concern. To utilize the output of a decision tree learner, we need to translate the decision tree into IF-THEN rules. However, the number of produced rules is as many as the number of leaf nodes in the tree. For a large tree with hundreds of leaves, hundreds of rules need to be hand-coded into the protocol. Therefore, we prefer to use the output from rule learners in implementing the online classifier, if they have comparable accuracy.

2. *TP rate* and *FP rate*: Given a classifier and an instance, there are four possible outcomes. If the instance is positive and it is classified as positive, it is counted as a *true positive* (TP). On the other hand, if the instance is negative and it is classified as positive, it is counted as a *false positive* (FP).

$$\text{TP rate} = \frac{\text{positives correctly classified}}{\text{total positives}} \tag{3.1}$$

$$\text{FP rate} = \frac{\text{negatives incorrectly classified}}{\text{total negatives}} \tag{3.2}$$

   It is crucial for us to consider the FP rate since the overall routing performance will suffer if we treat many low quality links as high quality ones. FP rate, therefore, is used here to represent the *cost* of learning. Usually we want a high TP rate (high benefits) and a low FP rate (low costs).

Additionally, because we need to use the classifier to guide link selection in collection routings, we prefer classification algorithms that produce human-readable outputs. The outputs from decision tree learners and rule learners can be used directly for this purpose. Other learning algorithms need extra tool-chains to transform their outputs into human-readable formats.

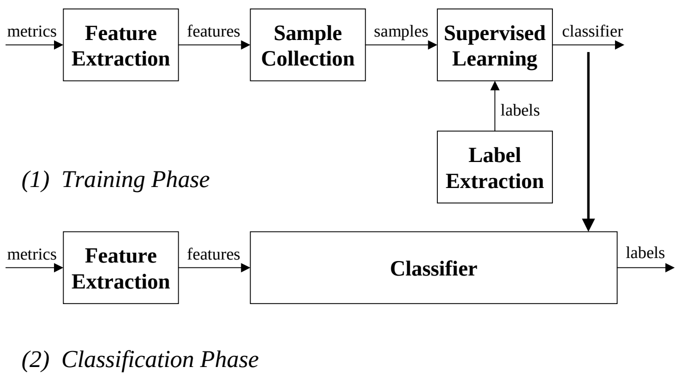


Figure 3.2: Overview of learning steps.

3.3 Learning Step-by-Step

In this section, we use link quality classification as the example to introduce the steps of our proposed learning method. Figure 3.2 presents a high level overview of the steps involved, with the four key steps listed as follows:

1. Feature extraction: In this step, we select the features to be used in training and classification.
2. Sample collection: Then, we instrument every node in the network to collect these features and their corresponding labels constantly and periodically send them back to the sink.
3. Training: Next, we used the labeled data to perform training at the sink node.
4. Classification: Finally, we instrument MintRoute to use the classifier for differentiating between high quality and low quality links at system running time. The algorithm is depicted in Section 3.4.

In what follows, we describe the first three steps listed above with specific reference to a collection routing application. Since the last step is closely related to the applica-

Link quality learning		
`RSSI`	received signal strength indication	local
`sendBuf`	send buffer size	local
`fwdBuf`	forward buffer size	local
`depth`	node depth from the base station	non-local
`CLA`	channel load assessment	local
`pSend`	forward probability	local
`pRecv`	backward probability	local

Table 3.1: Feature vector illustration.

tion, we put the discussion of application instrumentation using classifiers to Section 3.4. All results and analysis in this section are based on the data set from the case study—MetricMap—which is discussed shortly.

3.3.1 Step 1: Feature Extraction and Output Labeling

The first step in supervised learning extracts input features and labels output. This step requires domain knowledge to produce high-quality, well-prepared data [134].

In wireless sensor networks, we favor local features (within one-hop) that can be collected without expensive communications. This is because sensor networks are very resource constrained and it is desirable and necessary to impose as little overhead as possible. However, if a feature is already available with the existing routing protocol, such as node depth from MintRoute, we also consider it. There is no extra overhead required to gather this feature and it carries extra useful information.

Feature Selection

This is the process of choosing a subset of the feature space that best represents the problem at hand while introducing a minimal amount of noise.

As pointed out in previous studies, link quality is determined by many factors, including wireless channel conditions, such as internode separation, fast fading and slow fading, the traffic pattern in the network, and local traffic load of each node. However, the extent to which these factors impact link quality is continuously varying, which makes it impossible for any single metric to be always a good indicator of link quality. For example, [3] shows that SNR (Signal/Noise Ratio), though affecting link delivery probability, cannot be expected to be a predictive indicator of link quality. Thus, we choose a set of metrics correlated to link delivery probability to be included in the feature vector and use machine learning tools to train and identify the most predictive indicator, which could be a combination of them. Some of the metrics are related to channel conditions, some of them related to network congestion, and some of them to both. Table 3.1 lists the features we used for link quality learning and hop-by-hop caching learning. They are all numerical values.

RSSI is the received signal strength indication readily available in many commercial radios. It contains the average RSSI level during the reception of a packet by the CC2420 with its value appended to each frame. RSSI is averaged over 8 symbol periods (128μs) and is continuously updated for new symbols received. In the CC2420, LQI is a function of RSSI.

Channel load assessment is a metric used in CODA [126] to detect local network congestion. It uses a sampling scheme to monitoring local channel at appropriate times to minimize the energy cost while performing accurate estimates of congestion conditions.

Queue management is widely used in wired networks for congestion detection. In wireless networks, it is also closely related to local channel conditions. We use both forward buffer size and send buffer size as indications of congestion here. However, as pointed out in [126], without link-level acknowledgments, buffer occupancy or queue length cannot be used as an indication of congestion. In our experiment, link-level acknowledgment is enabled for the CC2420 radio.

Because the network topology may strongly influence the traffic load in a data funneling application, it could also have an impact on link delivery capability. Network topology can be characterized as node depth in a network or the number of children a node has in a collection tree. We use node depth, which is defined as the number of hops from this node to the sink in the collection tree. Due to funneling effects, node depth should be strongly correlated to link quality.

Lastly, $pSend$ and $pRecv$ are originally used to derive the average forward and backward delivery probability. Therefore, they capture important link quality information. On one hand, if their values are valid, they will contain history information of link delivery. On the other hand, if their values are invalid, they simply show the fact that something unexpected happened in the network, such as a congestion collapse, which could also be used to infer link quality. Therefore, we also include them as input features. We will show later in this section that these two metrics are very crucial in improving the classification accuracy.

Output Labeling

Output labeling is the process of classifying sample outputs using domain knowledge. Supervised learning algorithms need to use labels to determine the class to which the input features are assigned.

There are many ways to label link quality based on LQI. We study two approaches in this chapter. The first one uses a *binary* model that only predicts a link as "good" or "bad". The second one uses a *multi-class* model and can predict a set of classes of link quality. These link quality categories can be used to distinguish link quality in a finer granularity than using the binary model. To one extreme, the multi-class approach can predict the actual LQI numerically, which becomes a regression problem.

3.3.2 Step 2: Sample Collection

To perform the offline training, we need to collect samples from all nodes to the sink where learning is performed. However, transmitting samples by radio may interfere with the application traffic. If there is a programming board attached to each sensor node, we can access the samples directly from the network interface of the programming board, as configured in MoteLab. If there is no programming board attached, or if the sensor nodes are deployed in an environment where such a configuration is impossible, we can inject extra sensor nodes or some virtual sinks [127] that are used exclusively for siphoning the sample collecting traffic. In our experiment, we use programming boards to collect all samples.

Since link quality is strongly correlated with data traffic in the network, we collect samples from a variety of offered loads, ranging from 0.25 pps to 4 pps, in order not to lose traffic-related information. However, the number of samples collected from a non-congested network is far more than those collected from a congested network over the same sample collection period. Hence, we choose to use longer sample collection periods under high loads to guarantee that we have enough samples from a range of different loads.

3.3.3 Step 3: Offline Training

Our learning and validation experiment is performed on Weka [134], a workbench containing implementations of a variety of standard machine learning algorithms. We use the J4.8 algorithm provided with Weka for decision tree learning and JRip algorithm for classification rule learning. J4.8 implements an improved version of the C4.5 algorithm and JRip [26] implements Repeated Incremental Pruning to Produce Error Reduction (RIPPER), a propositional rule learner. C4.5 is one of the most widely studied and used decision tree algorithms in the literature. A thorough discussion can be found in [97].

As with most data-intensive machine learning algorithms, it is important to avoid having the classifier memorize, or overfit, the training data. We use cross validation and tree pruning in Weka to reduce such effects. Cross validation is a standard method to estimate classification accuracy over unseen data. We use 10-fold cross validation in our experiments. The available data is divided into ten equal-sized blocks. Nine of the blocks are randomly chosen and used for training a classifier, and the remaining block is used for validation. This process is repeated 10 times. The accuracy is 82% using J4.8 and 80% using JRip for our data set from MoteLab.

Table 3.2 shows the **confusion matrix** for a three-class prediction. Class a contains links with the best quality. Class c contains links with the worst quality. Class b contains links whose qualities are in between. A confusion matrix is often used to display the cost and accuracy of a multi-class prediction. Each element $(\mathbf{x}, \mathbf{y})$ in the matrix shows the number of samples for which the actual class is $\mathbf{x}$ and the predicted class is $\mathbf{y}$. The numbers down the main diagonal are those that are predicted correctly. The **accuracy** of our classifier is then $(1456 + 1369 + 1586)/5461 = 80.8\%$.

Table 3.3 shows the TP rate and FP rate of a three-class classifier for both JRip and J4.8, using the same link quality estimation dataset with 10-fold cross validation. For both algorithms, the FP rate of class c is lower than 5%, meaning that the probability of classifying a bad link as either a good or median link is low. In the context of metric-

```
rssi <= 212
|   depth <= 5
|   |   rssi <= 211: bad (320.0/37.0)
|   |   rssi > 211: good (79.0/34.0)
|   depth > 5: bad (425.0/31.0)
rssi > 212
|   rssi <= 223
|   |   cla <= 116
|   |   |   depth <= 3: good (352.0/82.0)
|   |   |   depth > 3
|   |   |   |   depth <= 4
|   |   |   |   |   rssi <= 220: bad (49.0/1.0)
|   |   |   |   |   rssi > 220
|   |   |   |   |   |   cla <= 8: good (69.0/29.0)
|   |   |   |   |   |   cla > 8: bad (14.0/4.0)
|   |   |   |   depth > 4
|   |   |   |   |   depth <= 6
|   |   |   |   |   |   rssi <= 216
|   |   |   |   |   |   |   depth <= 5: good (198.0/71.0)
|   |   |   |   |   |   |   depth > 5
|   |   |   |   |   |   |   |   rssi <= 214: bad (8.0/1.0)
|   |   |   |   |   |   |   |   rssi > 214
|   |   |   |   |   |   |   |   |   sendbuf <= 0
|   |   |   |   |   |   |   |   |   |   cla <= 21: bad (29.0/13.0)
|   |   |   |   |   |   |   |   |   |   cla > 21: good (2.0)
|   |   |   |   |   |   |   |   |   sendbuf > 0: good (2.0)
|   |   |   |   |   |   rssi > 216: good (178.0/34.0)
|   |   |   |   |   depth > 6
|   |   |   |   |   |   rssi <= 219
|   |   |   |   |   |   |   rssi <= 215: good (157.0/55.0)
|   |   |   |   |   |   |   rssi > 215
|   |   |   |   |   |   |   |   depth <= 7
|   |   |   |   |   |   |   |   |   rssi <= 217: bad (129.0/29.0)
|   |   |   |   |   |   |   |   |   rssi > 217
|   |   |   |   |   |   |   |   |   |   cla <= 0: good (20.0/6.0)
|   |   |   |   |   |   |   |   |   |   cla > 0: bad (12.0/3.0)
|   |   |   |   |   |   |   |   depth > 7
|   |   |   |   |   |   |   |   |   rssi <= 217: good (37.0/17.0)
|   |   |   |   |   |   |   |   |   rssi > 217
|   |   |   |   |   |   |   |   |   |   cla <= 0: bad (21.0/3.0)
|   |   |   |   |   |   |   |   |   |   cla > 0: good (2.0)
|   |   |   |   |   |   rssi > 219
|   |   |   |   |   |   |   depth <= 7
|   |   |   |   |   |   |   |   cla <= 3: good (102.0/35.0)
|   |   |   |   |   |   |   |   cla > 3: bad (30.0/12.0)
|   |   |   |   |   |   |   depth > 7: good (85.0/17.0)
|   |   cla > 116: good (62.0/8.0)
|   rssi > 223: good (275.0/38.0)
```

Figure 3.3: A sample decision tree output from Weka using a binary model for labeling. Each line represents one conditional branch in the tree. The pair of number (m/n) behind the label on each line means that there are a total of m instances that reach that leaf, of which n are classified incorrectly.

	Predicted class			
	a	b	c	Total
a (good)	1456	257	26	1739
b (medium)	403	1369	124	1896
c (bad)	86	154	1586	1826
Total	1945	1780	1736	5461

Table 3.2: Confusion matrix of a three-class classifier using JRip. Class a contains links with the best quality, and class c contains links with the worst quality. Class b contains the other links whose qualities are in between.

	JRip		**J4.8**	
Class	TP rate	FP rate	TP rate	FP rate
a (good)	0.837	0.131	0.841	0.133
b (medium)	0.722	**0.115**	0.712	**0.103**
c (bad)	0.869	**0.041**	0.885	**0.046**

Table 3.3: Detailed accuracy breakdown for all classes.

based routing, the cost of such mis-classification is high and both JRip and J4.8 work well in this aspect.

3.3.4 Discussion

Selection of learning algorithms. As we discussed earlier, the main criteria for selecting a learning algorithm are the learning accuracy, overhead, and cost. We have studied other classifiers, including ensemble methods, Bayesian classifiers, and regression methods, to get a feeling of the best accuracy we can achieve for this specific learning problem. From our experiment results, decision tree learners achieve higher accuracy in most cases than all the other methods we studied. Also, as introduced in Section 3.2, the output of decision tree learners is easy to interpret by human beings and the classification phase has low resource requirements. Therefore, we focus on decision tree learning for the following discussions.

Binary or Multi-class Classifier

One key difference between a binary classifier and a multi-class classifier is the flexibility in interpreting the labels. A link with median quality will either be classified as “good” or “bad” with a binary classifier. If the link is actually good enough but is classified as bad, potential good links will go unused. If we skew the threshold to treat more samples as good, the probability of not distinguishing between really good and fairly good links

	JRip		**J4.8**	
	Binary	Multiple	Binary	Multiple
Accuracy	82.6%	80.8%	85.2%	81.1%
Overhead	7 rules	16 rules	77 nodes	135 nodes
FP rate (bad)	5.9%	4.1%	11.3%	4.6%

Table 3.4: Comparison between a binary classifier and a multi-class classifier with three classes. FP rate is used to measure the cost of learning. A rule is an IF-THEN rule in JRip. The number of nodes is the total number of nodes of a decision tree in J4.8.

will occur. A multi-class classifier, however, will produce more information that can be leveraged.

The other extreme is a numerical classifier that predicts the exact LQI. However, classifiers like decision trees are accurate enough and efficient for our applications.

Table 3.4 compares the accuracy, memory footprint, and the false positive (FP) rate of class c (bad) between a binary classifier and a three-class classifier. Accuracy is defined as the percentage of instances correctly classified for all classes.

The accuracies of three-class classifiers for both JRip and J4.8 are lower than the accuracies of their correspondent binary classifiers by at most 3%. For J4.8, the size of the decision tree is relatively large. For JRip, however, the size increase is small since 16 rules usually take only several hundreds of bytes.

Feature Selection

Because irrelevant features will degrade the performance of decision trees and classification rules [134], it is beneficial to perform an attribute selection that will eliminate all but the most relevant features. We already selected a set of features based on our understanding of the problem domain and what each attribute actually means. Next, we use some well-established methods to further sieve these features to improve prediction accuracy or reduce the overhead of feature collecting.

M1		M2	
Rank	Feature	Rank	Feature
0.70812	psend	0.3251	RSSI
0.58138	RSSI	0.1577	fwd_buf
0.34003	precv	0.1384	psend
0.03586	depth	0.0771	precv
0.00406	fwd_buf	0.0628	depth
0	cla	0	send_buf
0	send_buf	0	cla

Table 3.5: Ranked attributes.

We use two attribute selectors provided by Weka, including InfoGainAttributeEval (M1) and GainRatioAttributeEval (M2) and use the union of the features ($M1 \cup M2$), as shown in Table 3.5.

M1 evaluates the worth of an attribute by measuring the information gain with respect to the class, while M2 measures the gain ratio. M2 takes into consideration the information each attribute contains that is neglected in M1. Equations 3.3 and 3.4 are the mathematical definition of the two metrics,

$$InfoGain(D, Attr) = I(D) - I(D|Attr) \tag{3.3}$$

$$GainRatio(D, Attr) = \frac{I(D) - I(D|Attr)}{I(Attr)} \tag{3.4}$$

where $I(D|Attr)$ is the entropy of training set D given attribute $Attr$ and $I(D)$ is the entropy of D. Entropy is widely used in machine learning to represent the amount of disorder an attribute contains with respect to the class of interest. In particular, $I(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$ where p_i is the probability that an arbitrary sample in D belongs to class C_i. Suppose samples in D on attribute $Attr$ having v distinct values as $\{a_1, a_2, \ldots, a_v\}$, then we have $I(D|Attr) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j)$ where D_j contains samples in D that have outcome a_j of $Attr$.

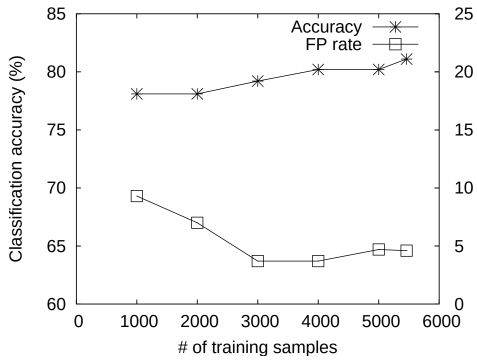
	7-feature	**5-feature**	**1-feature (RSSI)**	**1-feature (pSend)**
Accuracy	80.8%	80.8%	70.5%	69.3%
Overhead	16 rules	17 rules	4 rules	20 rules
FP rate (bad)	4.0%	4.1%	3.9%	4.1%

Table 3.6: Impact of feature selection. The 5-feature set is selected using the union of the features in M1 and M2 ($M1 \cup M2$).

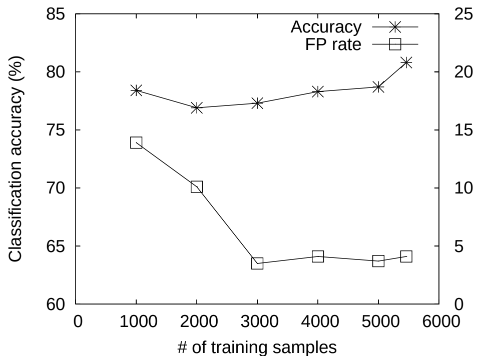
The impact of feature selection on the learning accuracy, memory footprint, and the FP rate of class c (bad) is demonstrated in Table 3.6. In particular, we compare the accuracy using all 7 features to the accuracy of using only one feature. Clearly, using more features results in a higher accuracy than using just one. This supports our motivation to study more features. As using 5-feature and 7-feature have a comparable classification accuracy, memory overhead and FP rate, we use the 5-feature set in this case study. It is interesting to note that the selection of features barely impacts the FP rate of class c (bad). One explanation is that both $RSSI$ and $pSend$ are critical in differentiating bad links from good ones. Including new features will not bring new information for this purpose.

Impact of Training Corpus Size

Figure 3.4 shows the impact of training corpus size on classification accuracy and FP rate. Empirically, 5000 samples are good enough for our application because incorporating more samples only brings marginal gains in improving learning accuracy and reducing FP rate.



(a) J4.8



(b) JRip

Figure 3.4: Accuracy (left-axis) and false positive rate (right-axis) as a function of the training corpus size.

3.4 Case Study

In this section, we present a case study to illustrate how supervised learning techniques can be leveraged to improve the performance of link-quality aware collection routing protocols in congested wireless sensor networks.

MintRoute is a collection routing protocol that uses ETX to construct routing topologies. As shown in Figure 3.1, MintRoute fails to find parents in congested networks, using snooping-based link quality estimation. However, if a parent can be identified based on other available information regarding link delivery capability, routing can be resumed and orphan nodes will be salvaged. We propose MetricMap, an alternative to MintRoute, that establishes link quality estimations using offline trained classifiers to address this problem.

MetricMap consists of two components. The first component controls the update of all features; it is triggered either by packet arrivals or timer events. The second component controls link classification, with input from features collected by the other component and output in numerical or categorical values indicating link quality. The output of the

```
// update feature vector on demand or periodically
void updateRSSI () {
  foreach packet successfully received from neighbor i
    keep the RSSI value history for i
}
void updateBuf (int type) {
  during each update interval
    update the buf size for type (fwdBuf or SendBuf)
}
void updateCLA () {
  during each update interval
    check the clear channel assessment and update CLA
}
void updateProbSend () {
  // this feature is updated the same as in MintRoute
}
void updateProbRecv () {
  // this feature is updated the same as in MintRoute
}
int classify (struct featureVec fv) {
  // perform classification based on input features
  // the output represents the class label
}
// update link quality based on classification results
// recvEst is the in-bound link quality estimation
// link quality is between 0 (low) and 255 (high)
void updateEst(fv) {
  if (classify(fv.rssi, fv.sendBuf, fv.fwdBuf, fv.depth,
      fv.CLA, fv.pSend, fv.pRecv) == "good") {
    recvEst = 1 * 255
  }
  else {
    recvEst = 0
  }
}
```

Figure 3.5: Pseudo-code of MetricMap.

classifier is used whenever the ETX-based method fails, which is detected whenever ETX returns an invalid value indicating the current node has no parent in the collection tree. We choose such a design in the consideration that if ETX is still working, it should give more accurate estimations of link quality than the offline-learned classifier. The pseudo-code of MetricMap is shown in Figure 3.5, with the function `classify()` implementing the second component and the rest implementing the first component.

3.5 Testbed Evaluation

To illustrate the application of supervised machine learning in realistic sensor network application settings, we have implemented the MetricMap prototype in TinyOS and evaluated its performance via a real-world testbed deployment of sensor nodes.

3.5.1 Evaluation Methodology

In our evaluation, we consider the following performance metrics:

- *Data delivery rate:* The fraction of data packets that are successfully delivered to the destination.

- *Data latency:* The time it takes from when a packet is sent out until the packet is received at the sink.

- *Fairness index:* This metric [52] is used to measure the variability of performance across all source nodes. For any given set of delivery rates $(p_1, \ldots, p_n)$, the fairness index definition adapted for our problem is given by:

$$f(p_1, \ldots, p_n) = \frac{(\Sigma_{i=1}^{n} p_i)^2}{n\Sigma_{i=1}^{n} p_i^2}$$

with p_i denoting the average packet delivery rate of the ith sensor and n the total number of source nodes in the network. The fairness index always lies between 0 and 1. If all nodes have the same packet delivery rate, the fairness index is 1.

In each experiment, we also measure the overhead required to achieve these performance metrics. In particular, we are interested in measuring the **memory footprint** of each protocol. The testbed is comprised of MicaZ motes which have the AT-

MEL 7.37MHz ATMega128L, low-power, 8-bit micro-controller with 128 KB of program memory, 512 KB measurement serial flash data memory, and 4 KB EEPROM. It uses a Chipcon CC2420, a single-chip IEEE 802.15.4 compliant Radio Frequency (RF) transceiver operating at 2.4 GHz and capable of transmitting at 250 kbps. The packet size we used in our experiments is 29 bytes, the default value in TinyOS. These motes are connected to an Ethernet used for logging and mote-programming.

3.5.2 MetricMap Results

We test MetricMap on the MoteLab testbed in the Harvard Computer Science Building [132]. It consisted of 30 motes across multiple offices at the time of these experiments.

Our experiment consists of two phases: the offline learning phase, which takes multiple hours for collecting training samples and processing the learning task using WEKA; and the online optimization phase that uses the rule-set learned in the training phase to guide situation-aware routing. Each run lasts 15 minutes for the routing performance to converge.

When we evaluate the performance of MintRoute and MetricMap on MoteLab, the results are different for runs at different times. This is because of uncontrollable factors in the testbed, especially the variability of link qualities. Therefore, we take the following approach to reduce the impact of uncontrollable factors in the environment. We run MintRoute followed by MetricMap or vice versa for a continuous 15 minutes. We run such pairs of experiment 5 times and each experiment is independent with respect to each other. Such a design allows us to reduce influences from factors other than the algorithm itself. Also, our experiments are performed both in the daytime and at night when the human activity interference decreases. For each offered load, the minimum, median, and maximum values are shown.

Performance and Overhead

Figure 3.6 compares the data delivery rate between MetricMap and MintRoute. Our approach consistently outperforms MintRoute. The higher the traffic load, the better MetricMap performs compared to MintRoute. MintRoute can rarely form a data collection tree under high traffic rates. In contrast, our approach can still form a tree because it leverages more information for link quality assessment.

Figure 3.7 shows the packet latency comparison. Packets delivered by MetricMap have a comparable average latency to those delivered by MintRoute. Data latency includes local processing time at the source node and all intermediate nodes along a multihop route, network transmission time over all links, and reception processing time at destination. Our classifier will be used regularly for updating the data collection tree. This may introduce some delay in the local processing time and transmission time if the calculation is on the critical path of data transmission. Our results show that the extra processing time in classification online does not impose a high overhead and delay on packet transmission.

Figure 3.8 compares the fairness index of packet delivery. It demonstrates that our approach is much better at maintaining fairness across different offered loads. It does not allow certain nodes to get unfair fractions of network bandwidth. This is reasonable since all nodes use similar rule-sets learned offline and there is no bias towards any particular link. On the other hand, since MintRoute relies on data traffic to infer link quality, the link selected may be skewed depending on the traffic pattern and the node's location to the sink. If any part of the network en route to the sink is overloaded, the MintRoute

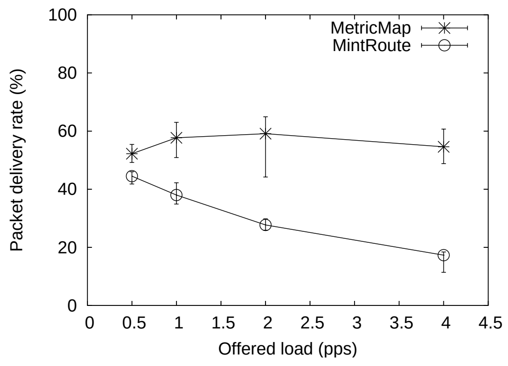


Figure 3.6: Average success rate versus per-sensor load using a periodic workload.

data collection process will be interrupted. MintRoute uses broadcast in this case to try to resume the communication, but this actually exacerbates the problem by adding more useless traffic into the network. Our classifier can mitigate the problems by discerning meaningful link information without imposing any additional traffic. Once the routing tree is re-formed, the data collection process can be resumed very quickly. So, using MetricMap, more nodes can deliver their data to the sink, which results in a higher fairness index. In contrast, using MintRoute, a few nodes deliver many packets and the rest have a very low success rate.

In summary, MetricMap addresses the high data rate challenge from a different perspective compared to congestion control mechanisms [49, 126, 127]. Namely, we use a range of parameters to guide link selection for success and fairness in data delivery. Our approach is expected to be orthogonal to theirs and combining them could potentially achieve further performance improvement.

Since MetricMap needs to keep local metrics that are used as input to the classifier, it requires some extra memory usage. We use the memory footprint of MetricMap to quantify the overhead. Table 3.7 shows the actual memory footprint of MintRoute and MetricMap. The increase in program size is 11.5%, which is used mostly for imple-

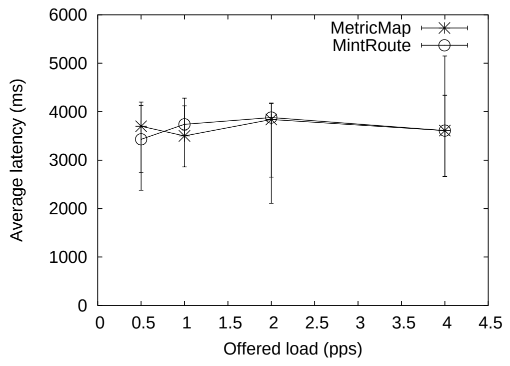


Figure 3.7: Average packet latency versus per-sensor load using a periodic workload.

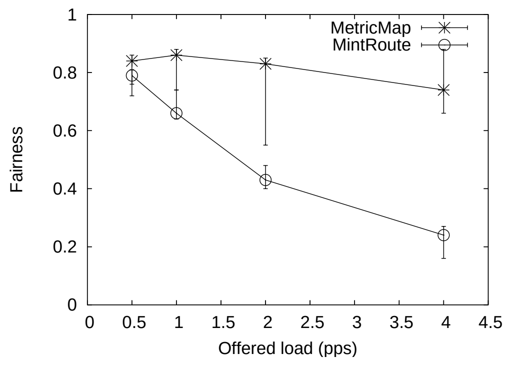


Figure 3.8: Fairness index of packet delivery rate versus per-sensor load using a periodic workload.

menting the classifier. The increase in static memory size is 7.1%, which is mostly data structures used for collecting and converting low-level metrics to input of the classifier. This is a small increase from the original code and memory footprint.

Other than memory cost, there are additional learning costs in the sample collection and training phases. It would be interesting to determine if these costs can be amortized to many weeks of protocol running without re-training; we leave this as future work.

Our results so far have shown that MetricMap produces consistently higher performance than MintRoute under heavy traffic load. To understand if such benefits come from a better selection of good quality links, we further compare MetricMap with another data

Component	**ROM (Flash)**	**RAM**
Surge+MintRoute	16570	1971
Surge+MetricMap	18468	2110

Table 3.7: Code and memory usage comparisons of MintRoute and MetricMap on MicaZ. RAM is memory usage in bytes and ROM is program size in bytes.

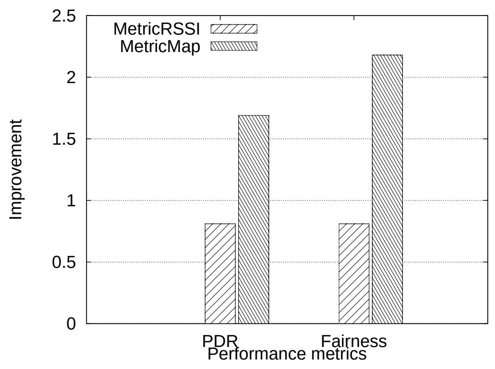


Figure 3.9: Performance improvement comparison with heuristics-based approach.

collection protocol — MetricRSSI. MetricRSSI uses the RSSI values of received packets over a link as the only indication of its quality. If the recently received packets have higher RSSI values compared to other links, the protocol will assign a higher quality value to this link than other ones. Other than that, MetricRSSI is the same as MetricMap. Thus, MetricRSSI does not take into account any factors other than packet RSSI values and makes its estimation solely using heuristics.

Figure 3.9 shows the average improvement of MetricRSSI and MetricMap over 5 independent testbed runs, using the performance of MintRoute as the base line. For example, the improvement of protocol X in packet delivery rate is calculated as $(p_X - p_{MintRoute})/(p_{MintRoute})$. The figure shows that MetricMap has higher packet delivery rate and fairness index compared to MetricRSSI. Because MetricMap uses more features to make its link quality estimates, it potentially will find better links that have the capability to deliver more traffic.

There is a minor increase in data latency for both protocols. This is because both MetricRSSI and MetricMap deliver more packets than MintRoute and these packets usually have more hops to traverse. Since the increases of data latency for both protocols are negligible, we do not discuss them further.

3.6 Online Learning

Extensive empirical studies of real-world sensor networks have shown that high variability in the quality of radio communications exists between low power sensor devices. Therefore, it is necessary to continuously take new data into account and adjust learning results. The offline learning approach proposed in previous sections can be used periodically to adapt to such changes. However, this approach has two limitations. First, it uses traditional decision tree learners that need to store the entire data set (training samples) for training. Second, it involves transferring all training sets to the base station, which is a huge burden to the underlying network infrastructure.

On the contrary, online learners work incrementally as new data is received over time. There is no need to store the entire data set for training purposes since data samples are treated as a *data stream*. This makes distributed learning feasible in sensor networks since some aspects of the learning process such as data sampling, aggregation, and training can now be placed on individual, storage-constrained sensor nodes or master nodes [38], such as Stargates. This also eliminates the problem of periodically collecting new samples to the base station.

In this section, we explore the possibility of using online learning methods to maintain efficiency and accuracy, while being able to quickly adapt to changing environments. We focus on online, incremental decision tree learners due to their output interpretability and learning accuracy.

3.6.1 Online Learning Overview

In many problem domains, the information required for learning is rarely available *a priori*. With new pieces of information becoming available over time, the decision structures should be revised as necessary. Such a learning mode is identified as incremental learning, or online learning.

The VFDT (Very Fast Decision Tree) [29] learner is an online learning algorithm that manages stream data using few computational resources, while maintaining a performance similar to traditional batch learners. In VFDT, a decision tree is learned by recursively replacing leaves with decision nodes when new samples are available. Each leaf node stores the statistics about attribute values that are used to measure the merit of split-tests. VFDT uses Hoeffding trees, which exploit the Hoeffding bound (or additive Chernoff bound) to determine, with high probability, the smallest number of samples necessary at each leaf to select a splitting attribute that would be the same as one chosen using the entire data set. One attractive property of the Hoeffding bound is that it is independent of the probability distribution generating the observations.

3.6.2 Evaluation

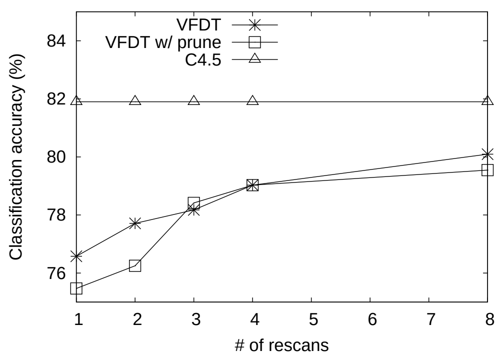
Methodology and Data Set

We use the VFDT implementation provided in the VFML (Very Fast Machine Learning) toolkit [123]. The original design of VFDT is targeted for effective learning from very large data sets, such as web-click stream data. In our case, since link quality data is continuously generated, the learning problem fits naturally into the stream data domain. However, the number of link quality samples generated is usually much smaller than the number generated by other data sets used in the original VFDT implementation. This is because link quality data collection depends on the traffic rate and the topology of the testbed. To take this factor into consideration, we enable the **-rescans** option in the VFDT reference implementation. Activating this option allows VFDT to rescan previously-seen samples, which helps to gather statistics of attribute values and improve classification accuracy. This is important for small data sets or in situations in which data arrives slowly. However, this is only an artifact of our data set. Since sensor networks are designed to operate for months or even years, we anticipate that a large volume of new data samples will be available and that makes the online incremental learning approach more feasible. Rescanning allows us to compare between VFDT and C4.5 in a fair way.

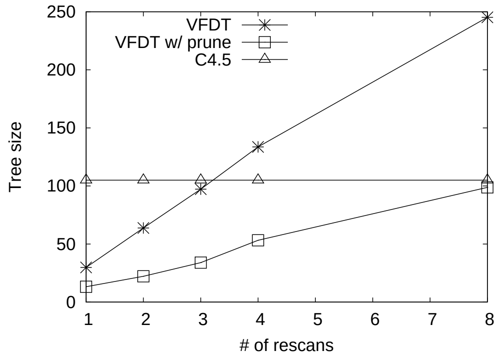
The data set we used for evaluating the accuracy of VFDT is the same as we used in Section 3.3 for traditional batch learners. In particular, we compare the classification accuracy of VFDT with C4.5 [97]. We still used 10-fold cross validation to get the accuracy results. We also used shuffled data sets of the original one to produce different data streams so that we could test the sensitivity of VFDT to dynamic link quality data. Since the tree size produced is of special interest in the context of sensor networks, we also tested the difference of accuracy and tree size between a pruned tree and a non-pruned tree.

Evaluation Results

We conducted a series of experiments to evaluate the classification accuracy of VFDT as a function of the number of rescans. With the technology advances in storage and memory devices, we can imagine that sensor nodes in the near future will be equipped



(a) Classification accuracy



(b) Tree size

Figure 3.10: Performance as a function of the number of rescans.

with even more memory space than the Megabytes of Flash memory they currently have. If we adopt a hierarchical architecture [38], the master nodes usually have more memory space than other nodes. Therefore, it is feasible for the sensor nodes or master nodes to store recently-seen data sets for rescan. In our experiment, we stored about 3K samples. Figure 3.10(a) shows that increasing the number of rescans improves the classification accuracy. The tradeoff here is that increasing the number of rescans also increases the tree size created, which is reflected in Figure 3.10(b). A larger tree leads to a more complicated classifier and more computation processing during link quality classification. This could be a design option based on the available memory, the desired classification accuracy, etc. Further exploration of such a tradeoff is left as future work.

Table 3.8 shows the classification error rate and tree size using shuffled data sets of the original link quality data collected from the MoteLab. Data samples are shuffled in a purely random way to eliminate potential correlations in the original data set. The results show that VFDT is robust to a dynamic data stream and maintains comparable performance across all data sets. We also see that the pruned trees are more desirable in our case because they produce a classification accuracy similar to that of VFDT without pruning and that of C4.5 with a much smaller tree size.

Shuffled	**C4.5**		**VFDT** (rescan:4)		**VFDT w/ prune** (rescan:4)	
dataset	Error (%)	Tree size	Error (%)	Tree size	Error (%)	Tree size
1	18.10	105	20.97	134	20.97	53
2	18.64	107	21.90	130	22.16	44
3	18.21	102	20.62	130	21.56	49
4	19.60	105	21.65	134	20.93	50
5	18.37	104	19.68	129	21.06	48
6	18.45	107	20.54	137	21.39	49

Table 3.8: Learning results with shuffled datasets.

In summary, online learning methods produce classifiers with similar accuracy to traditional batch learners. Since they do not need to work on the entire data set, they are more resource- and computation-efficient. Our preliminary evaluation shows that online learning provides a promising approach for learning tasks in wireless sensor networks.

3.7 Related Work

Significant work has focused on providing nodes with the ability to rapidly observe and react to the dynamics in wireless sensor networks, where a wide range of network conditions exist. The awareness of network situations would allow a more adaptive protocol to be deployed. Most of the previous work either uses a “rule of thumb” focusing on a single metric that may lose useful information or even worse, lead to misunderstanding of situations, or uses sophisticated heuristics that combine a lot of parameters and require a lot of expertise and domain knowledge to derive. This section surveys related work on situation-aware routing in wireless sensor networks and also reviews applications of machine learning to problems in other domains.

Situation-awareness. Debugging and diagnosis also focus on finding problems and providing information. Sympathy [98] is developed as a tool-set for detecting and debugging failures in sensor networks. Sympathy is mainly used as an automatic debugging tool for root cause detection of failures in a centralized location, such as the sink. SNMS [118] is another network management system for wireless sensor networks. However, the focus of SNMS is to facilitate the network management for operators. Our focus, instead, is to provide knowledge as to which metric, or what combination, should be used in adaptation to network dynamics, using standard machine learning and data mining technique.

Machine learning. There has been significant prior work on applying machine learning techniques to different areas of research, and most recently system-related problems, such as compiler optimization [15], system performance diagnosis [25], fault localization in Internet services [24], and software bug isolation [69].

Various machine learning techniques have been used in dynamic routing optimizations, such as reinforcement learning [9], and Bayesian inference [87]. Since our proposed optimization works by identifying good links in congested networks, we focus on classification problems in this chapter, which is particularly suitable for algorithms such as decision trees and induction rules. It is interesting to investigate the effectiveness of other learning methods for routing optimizations in wireless sensor networks and we leave it as future work.

Machine learning has also been used for modeling data generated by sensor networks. Guestrin *et al.* [40] used kernel-based regression to accurately model sensor data and reduce the dimensionality of data representation. This approach significantly decreases the communication requirements in the network. More recently, Krause *et al.* [66] studied sensor placements using probabilistic models that account for both data quality and communication costs. Our approach, however, focuses on optimizations within the networking protocol stack.

Link quality estimation. Link quality awareness permeates many aspects of sensor network design and operation [94], ranging from the design of MAC protocols to the design of applications. As a result, link quality estimation has become a significant research focus in recent years.

One challenge with link quality estimation in sensor networks is to maintain accurate and reliable estimations with low overhead; many improvements have been aimed at the original ETX approach. Many of the proposed metrics [21, 22, 64, 139], however, share one limitation: The performance of their metrics depends heavily on their model accuracies, which need significant trial-and-error tuning and expert knowledge. Our approach, on the other hand, passively collects features that are readily available and uses standard learning algorithms to discover the inner correlation. Furthermore, their observations on temporal and spatial variability of channel conditions can be used in our work to improve learning efficiency. For example, the temporal variability can be used to determine the interval for relearning. The spatial variability can be used to perform distributed learning, instead of collecting all samples to a central location.

A similar problem to link quality estimation is lossy links identification. Nguyen *et al.* [86] proposed to use end-to-end data for lossy links inference. Our technique can also be used for this purpose. Furthermore, since our approach is not limited by the end-to-end data assumption, intermediate nodes can also infer lossy links. This makes it feasible for our approach to be used for routing optimizations, while their approach is focused mostly on inferring lossy links.

Routing optimization. In terms of efficient routing design in the presence of unreliable radio links, [17] takes a joint-optimization approach that considers both the recovery of lost packets in the link layer as well as path selection in the routing layer. The metric they proposed considers many of the features we use in this work. However, our focus is on learning information that is otherwise unavailable with traditional approaches. Therefore, our method can be combined with theirs to further improve communication efficiency.

3.8 Conclusions and Future Work

This chapter presents a supervised learning framework that can be used to produce useful information automatically and to help make informed decisions in sensor networks. As a case study, we investigated the link quality estimation problem by casting it into a classification problem. Experiments on a real-world sensor network testbed show that our technique can achieve significant performance improvement over existing approaches. We also explored the possibility of using online learning algorithms to efficiently adapt to external changes and varying network conditions. Our results show that the online learning algorithm we used achieves similar accuracy compared to traditional batch learners, but is more resource- and computation-efficient.

Beyond this initial prototype, we envision future work to include the following. First, one could further test the effectiveness of VFDT on real-world testbeds by embedding its output classifier into MetricMap. Second, there may be significant external changes that will affect the correlation between link quality and other parameters, which we term as "concept shifting". We are interested in studying online learners that are capable of detecting and capturing such concept-shifts. Third, we wish to apply our learning techniques to new problems that may benefit from our approach, such as root cause analysis of packet loss [112].

Overall, this work offers an important first look at machine learning techniques for the particular network problems we have evaluated. In demonstrating machine learning's considerable performance advantages, this work has made a first step towards clean implementations of highly-effective, situation-aware learners for real-world sensor networks and other similar networks.

Chapter 4

Erasure-Coding-Based Routing for Opportunistic Networks

In previous chapters, we discussed ways of achieving situation-awareness by exposing new knowledge to traditional protocols to improve performance. The basic operations of the protocols were kept untouched as much as possible, while the decision-making components were replaced with our situation-aware ones. This approach, however, may not always be feasible. For example, in many sparsely-connected or intermittently-connected networks, traditional MANET protocols such as DSR and AODV will not work well even equipped with the knowledge of network disconnection, since they seek to establish end-to-end paths between the communicating pairs. In these situations, new solutions are required to make routing efficient and resilient to the impact of disruptions.

In the following two chapters, we propose two such new solutions to combat disruptions where few traditional approaches apply. While disruptions in challenged networks could be caused by a wide array of factors, one major type of disruption is related to extreme mobility, such as that found in many DTNs and opportunistic networks. In this chapter, we investigate the performance problems facing current routing protocols in opportunistic networks and propose an erasure-coding-based routing protocol to achieve guaranteed low message delivery latency.

4.1 Introduction and Background

Routing in mobile sensor networks with unpredictable mobility is a challenging task because disconnections are frequent and the lack of knowledge about network dynamics hinders accurate decision-making. Existing approaches primarily use redundant transmissions to achieve reliability, which have either high overhead due to excessive transmissions or long delays due to incorrect choices during forwarding under energy budget (only a few redundant copies are allowed). Instead of predicting the best relays for forwarding, we propose a novel forwarding algorithm based on the idea of erasure coding to reduce the dependency on mobility and increase the resiliency to mobility dynamics. The key benefits of forwarding code blocks lie in that we use more relays for data forwarding to mitigate the impact of outlier forwarders. This scheme works even when only a subset of the relays successfully deliver their data. Furthermore, our approach maintains a constant overhead that is limited by the energy budget and adds no extra data transmission overhead compared to approaches using redundant transmissions.

Next, we provide a general overview of the state-of-the-art communication methods for challenged mobile networks, including DTNs and opportunistic networks. We focus on the tradeoffs and constraints that lead to our design decisions in this chapter.

4.1.1 Delay/Disruption Tolerant Networks

Delay/Disruption Tolerant Networks have recently emerged as an important area of network research that cover many of the performance-challenged networks mentioned in this dissertation. The DTN architecture [28] is proposed as an approach for achieving efficient communication in DTNs. It adopts the store-and-forward routing paradigm to handle disruptions, such as frequent disconnection, extremely long delay, and high bit-error rate. Due to intermittent disconnection, frequent network partitions and other disruptions, no end-to-end contemporaneous routes exist between the source and destination. Data is buffered at intermediate nodes during periods of disconnection or long delays to achieve resilience to disruptions. This reflects a paradigm shift from data- or communication-centric approaches to storage-centric approaches to achieve reliable communication in many challenged mobile networks.

The study of store-and-forward asynchronous communications in poorly connected networks has a long history. UUCP (Unix to Unix Copy) [121] and FidoNet [35] are early examples of such messaging systems based on the store-and-forward paradigm. Wizzy Digital Courier [135] is a project to distribute data, such as emails, to places with no Internet connection, using UUCP as its major communication protocol. However, these earlier systems work with only a few lower level technologies. On the contrary, the DTN architecture provides a general solution to accommodate very different lower level technologies and support interoperability across radically heterogeneous networks.

The DTN architecture is designed to allow nodes to communicate even when partitioned from each other for extended periods of time. It specifies how a set of DTN routers form an overlay network to cooperatively store and forward bundles of data. DTN routers are connected by links that can be persistent, scheduled, or opportunistic. The DTN architecture has been widely used in performance-challenged networks to address their routing difficulties. Routing algorithms for DTNs are not included as part of the architecture itself. This is because routing in DTNs is concerned about eventual delivery by employing long-term storage at the intermediate nodes. Therefore, routing algorithms are focused on locating when and to whom to relay messages, which are highly dependent on the time-varying characteristics of the network and the availability of such information.

To address this open problem, a number of routing algorithms for DTNs have been proposed [53, 57, 70, 122]. Generally speaking, current approaches can be divided into two classes: those based on epidemic message replications, and those based on knowledge of contact schedules. Epidemic routing [122] is one of the early proposals for routing in partially connected networks. They introduce random pair-wise message exchanges among mobile nodes to achieve eventual message delivery. Their goal is to maximize message delivery rate, minimize message delivery latency and the aggregate system resources. To achieve this, tradeoffs must be made. They choose to place an upper bound on message hop counts and per-node buffer space to accomplish this. Epidemic routing is well-suited to networks that need to self-organize, since they do not rely on pre-known knowledge of node contacts. Unfortunately, it may suffer from limited buffer space and energy budget (which determines the number of copies allowed in the network).

The algorithms proposed in [53] assume the availability of contact schedule knowledge. In [53], the general DTN routing problem is formalized and framed as a constrained optimization problem in which network links may go on and off for extended periods of time and each node has a storage constraint. Routing algorithms are further divided into three classes, depending on the availability of knowledge: routing with zero knowledge, with complete knowledge, and with partial knowledge respectively. They show that global knowledge may not be necessary to achieve good performance and smarter algorithms may provide a significant benefit in resource-constrained conditions in terms of contact opportunities, bandwidth or storage. However, they do not discuss how to leverage partial or local knowledge effectively to achieve good performance; this merits further investigation, since many existing challenged mobile networks only have partial or local knowledge to make routing decisions. To address this limitation, [57] presents a routing protocol that only uses observed information of the network. They use metrics to estimate the utility of next hops in terms of certain performance goals, such as minimum delay and provide information for routing decision makings. This is in essence very similar to our approaches in Chapter 2. However, acquiring network knowledge introduces bookkeeping overhead and extra communication, which may interfere with normal message transmissions. Furthermore, no accurate knowledge may be easily obtained for networks with high unpredictability.

4.1.2 Opportunistic Networks

Opportunistic networks [89] evolved from both MANETs and DTNs. In opportunistic networks, nodes make no assumptions about the existence of end-to-end paths from the source to the destination. Contacts between nodes are usually neither persistent nor scheduled. This is in stark contrast to traditional MANETs that assume end-to-end connectivity. Routing in MANETs, therefore, is about route discovery and maintenance when mobility causes churns (changes in membership) in the routing table. Routing in opportunistic networks is in essence a prediction problem because the decisions as to when and to whom to forward data are the most critical. The concepts of routing and forwarding are intermixed since routes are built during data forwarding. The nature of opportunistic networks suggests that long latency may be required for eventual data delivery. Therefore, they also fit under the category of DTNs. Examples of opportunistic networks include ZebraNet, Pocket Switched Networks [47], and Shared Wireless Inforstation Model (SWIM) [110].

Other than the DTN routing algorithms discussed in the last section, several new routing techniques have been proposed specifically to resolve mobility problems. Message Ferrying [146] investigates the usage of proactive mobility to meet communication needs. With nodes actively scheduled, contacts in such networks can be treated as partially scheduled and partially opportunistic. However, due to the requirements of having such high end proactive mobile equipments, such networks involve new energy and performance tradeoffs. Data MULEs [108] are another example; they leverage the opportunities from node mobility for data delivery. These are interesting directions to investigate. However, our solutions presented in this dissertation assume no such controllable mobility.

Recent Trends. Additionally, projects such as Urban Sensing [13], MetroSense [16], and SensorPlanet [106] have recently explored sensor networks that cover very large areas, such as an entire city. These networks span the areas of mobile ad hoc networks and DTNs and are confronted with a hybrid category of challenges from both domains. As a result, the discoveries in this dissertation can contribute to this emerging field as well.

4.2 Routing Algorithms Classification

In this section, we review the designs of forwarding algorithms that have been proposed especially for dealing with intermittent disconnections in DTNs and opportunistic networks in particular. We will use those algorithms later in this chapter for performance evaluation against our proposal. These algorithms differ in their decisions as to *who* forwards the data, *at what time* is the data forwarded, and to *whom* is the data sent. In the following discussions, we define a *contact* as an opportunity to communicate between two nodes and a *relay* as a forwarding node.

- *Flooding (*`flood`*):* In flooding, each node forwards any non-duplicated messages (including messages received on behalf of other nodes) to any other node that it encounters. It delivers messages with the minimum delay if there are no resource constraints, such as link bandwidth or node storage. We use `flood` as the reference for the best achievable performance in terms of data delivery rate and data latency.

- *Direct contact (*`direct`*):* In direct contact, the source holds the data until it comes in contact with the destination. Therefore, `direct` uses minimal resources since each message is transmitted at most once. However, it may incur long delays.

- *Simple replication (*`srep(`r`)`*):* This is a simple replication strategy in which identical copies of the message are sent over the *first r contacts* . Here, r is the replication factor determined by the energy budget. Only the source of the message sends multiple copies. The relay nodes are allowed to send only to the destination; they do not forward it to other relays. This leads to small overhead as the message flooding is controlled to take place only near the source. This class of forwarding algorithms is also known as the *two-hop* relay algorithm [39, 47]. There is a natural tradeoff between overhead (r) and data delivery latency. A higher r leads to more storage/transmissions, but a lower delay.

- *History-based (*`history(`r`)`*):* Here *history* is used as an indicator of the probability of delivery. Each node keeps track of the probability that a given node will deliver its messages. The r highest ranked relays (based on delivery probability) are selected as forwarding nodes. ZebraNet uses the frequency at which a node encounters destination as an indicator of the delivery probability. We use the implementation described in [58] in this study. One thing to note, source nodes do not store copies of messages that they generate, after enough copies of the messages have been forwarded. This explains why the `history` protocol sometimes performs worse than `direct` in our simulation.

A summary of the above forwarding algorithms is listed in Table 4.1.

Table 4.1: Forwarding Algorithms

Algorithm	Who	When	To whom
`flood`	all nodes	new contact	all new
`direct`	source only	destination	destination only
`srep(`r`)`	source only	new contact	r first contacts
`history(`r`)`	all nodes	new contact	r highest ranked

4.3 Solution Overview

In this chapter, we propose our solution to energy-efficient routing in opportunistic networks, using the ZebraNet mobility as the evaluation model. As we have discussed, routing in such networks cannot rely on setting up an end-to-end path from the source to the destination, because contact dynamics are not known in advance and such a path may not exist at all. Most current approaches are based on message replication over multiple contacts [58, 122], which is limited by the energy budget and available buffer space. To increase message delivery latency, more copies of message need to be forwarded in the hope that some of them may deliver the data sooner, which in turn increases the energy consumption. Given a specific energy budget in terms of the number of messages allowed to forward, the delivery latency may suffer if the selected relays do not meet the destination soon.

We propose an alternate method to improve message delivery latency performance, while keeping the overhead (the number of bytes transmitted at each node) fixed. The basic idea is to erasure code a message and distribute the code blocks over a larger number of relays than replication-based methods. Compared to sending a full copy of the message over a relay, only a fraction of code blocks are sent over each relay. Thus, we can control the routing overhead in terms of bytes transmitted, while forwarding to more relays at the same time. For scenarios like ZebraNet, where nodes are energy constrained, limiting such overhead is an important design goal.

The basic idea of using erasure coding is simple and has been explored in many areas [82]. However, it is not clear if and when it will perform better than simpler alternatives based on pure replications in DTNs. In this chapter, we conduct performance comparison between the erasure coding approach and the other alternatives using a variety of mobility scenarios with different node densities. We use both synthetic and real-world DTN mobility traces as input to our simulations. We discover that the erasure coding approach can provide good delay guarantees by using a fixed overhead. Fundamentally, the benefits of erasure coding arise in eliminating cases when long delays arise due to bad choice of forwarding relays. Erasure coding allows the transmission to be spread over multiple relays while using a fixed amount of overhead. This results in a protocol much more robust to failures of a few relays or some bad choices. We find that the erasure-coding-based algorithm is the least sensitive to different parameters in terms of message latency and message delivery rate.

4.4 Erasure-Coding-Based Forwarding Algorithm

As discussed in the previous section, most current approaches for routing in opportunistic networks are based on sending multiple identical copies over different paths. There is a fundamental tradeoff between overhead and delay. On one extreme, flooding achieves the best possible delay but results in very high overhead. The other extreme is protocols like `direct` which have low overhead because they send only few copies or none at all. The lack of knowledge about the topology dynamics inhibits distinguishing good paths from bad ones. Therefore, these protocols may result in long delays if bad paths are selected. In this section, we present a forwarding algorithm based on the idea of erasure coding. Our algorithm achieves better worst case delay performance than existing approaches with a fixed overhead.

4.4.1 Erasure Coding Basics

Erasure codes operate by converting a message into a larger set of code blocks such that any sufficiently large subset of the generated code blocks can be used to reconstruct the original message. More precisely, an erasure encoding takes as input a message of size M and a replication factor r. The algorithm produces $M * r/b$ equally sized code blocks of size b, such that any $(1 + \epsilon) \cdot M/b$ erasure-coded blocks can be used to reconstruct the original message. Here, ϵ is a small constant and varies depending on the exact algorithm used, which could be Reed-Solomon codes [100] or Tornado codes [73]. The selection of algorithms has to consider the tradeoffs between coding/decoding efficiency and the size of the code blocks generated. For example, Tornado codes have efficient encoding and decoding steps based on simple operations such as XOR, at the cost of slightly higher ϵ. A thorough discussion of the various tradeoffs is presented in [82]. The choice of which erasure coding algorithm to use is not the focus of this work. The key aspect is that when using erasure coding with a replication factor of r, only $1/r$ of the code blocks are required to decode the message. Hereafter, we ignore the constant ϵ for simplicity.

4.4.2 Erasure Coding Based Forwarding (`ec`)

Our erasure-coding-based forwarding algorithm can be understood as an enhancement to the simple replication algorithm (`srep`) described in Section 4.1.

In `srep` with a replication factor r, the source sends r identical copies over r contacts and relays are only allowed to send directly to the destination. In the erasure-coding-based algorithm, we first encode the message at the source and generate a large number of code blocks. The generated code blocks are then equally split among the first kr relays, for some constant k. In comparison with `srep`, this approach uses a factor of k more relays and each relay carries a factor of $1/k$ less data. However, the number of bytes generated are rM, the same as the number of bytes generated by `srep` (r).

By definition of erasure coding (with a replication factor of r and a message size of M), the message can be decoded at the destination if $1/r$ of the generated code blocks are received. Since code blocks are divided equally among kr relays, the message can be decoded as soon as any k relays forward their data, assuming no code blocks are lost during transmissions to and from a relay. When $k = 1$, the erasure coding approach is reduced to the simple replication approach, which is to use the first r relays to carry copies of the original message.

In simple replication, r relays are used to improve the delay performance. The erasure-coding-based approach, instead, utilizes kr relays for the same amount of overhead. Therefore, one can expect that the chances of at least some relays having low delays are higher, compared to using only r relays. At the same time, erasure coding requires at least k relays to succeed (instead of 1 in `srep`) in order to reconstruct the original message. Therefore, if the number of such low-delay relays are equal to or larger than k, the erasure-coding-based approach will successfully deliver the message with a lower delay than using simple replication. In a sparse network, it might have to wait a long time to get all these relays though.

4.5 Evaluation

In this section, we use simulation to compare the four forwarding algorithms described in Section 4.1 and our erasure-coding-based approach.

4.5.1 Methodology

We use `dtnsim`, the discrete event simulator for DTN environments from [53]. We implemented the following routing algorithms in `dtnsim`: flooding (`flood`), direct contact routing (`direct`), history-based routing (`history`), simple replication routing (`srep`) and erasure-coding-based routing (`ec`). For `srep` and `ec`, we represent different replication factors and number of relays used to split, using `srep`-rep$r$ and ec-rep$r$-pn. Here, r is the replication factor and n is the number of relays among which code blocks are divided.

We simulate using the same real-world mobility trace described in Section 2.5.1. We scale the grid size to 6km$\times$$6$km with a radio range of 1km. Initially, the nodes are randomly distributed in the grid. The base station moves along a rectangular path near the grid boundary. All messages are of size 1M. Each node generates 12 messages every day. The total duration of simulation is 16 days.

We are interested in the following performance metrics:

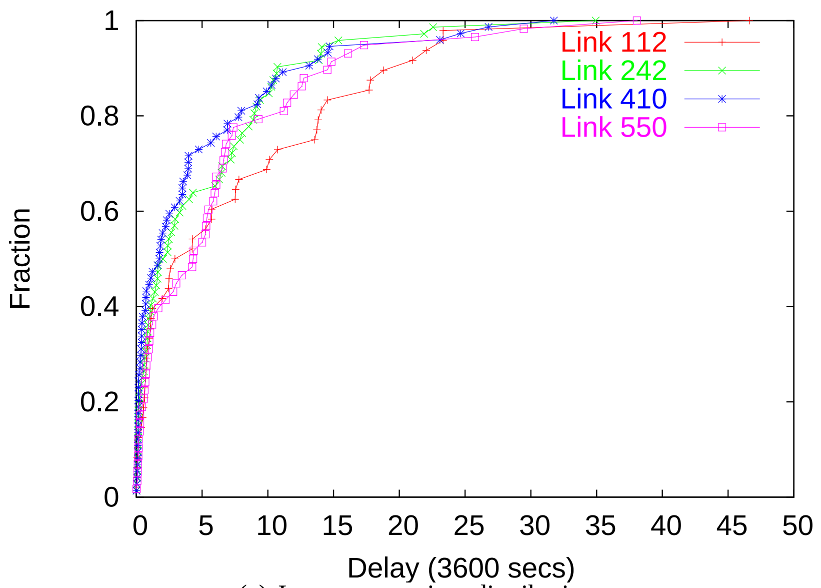
- *Data success rate:* The ratio of the number of messages that are delivered to the total number of messages generated within a time T (**deadline**). If T is unspecified, it is considered to be the whole duration of the simulation, i.e 16 days.

- *Data latency:* The duration between message generation and message reception (at its destination). In a DTN, latency may not be the most critical issue. However, it is always desirable to have fast data delivery whenever possible. The latency distribution metric measures how efficiently a protocol uses the available contact opportunities.

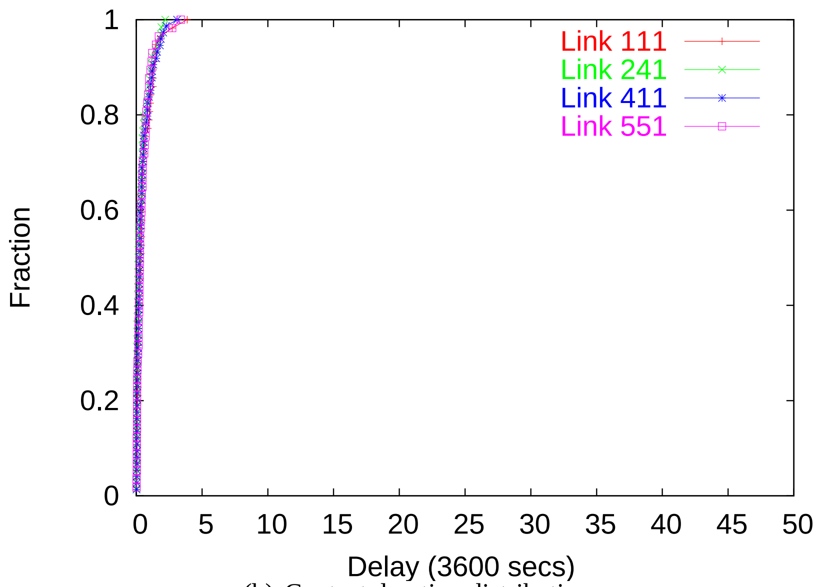
- *Routing overhead:* The ratio of the number of bytes transmitted to the number of bytes generated during the simulation time. This metric measures the extra data transmitted for each message generated, while a metric based solely on the number of message transmissions will overlook the fact that `ec` has smaller message sizes. The radio transmission energy is proportional to the total number of byes transmitted. Therefore, this metric reflects the energy efficiency of the forwarding algorithm.

4.5.2 Zebra Trace Analysis

To begin our analysis, we first characterize the contact opportunities in the ZebraNet trace, with a focus on inter-contact time and contact durations. These two metrics are important in understanding the behavior of different forwarding algorithms on the ZebraNet trace. Simply put, inter-contact time is the time interval for which a link is down (no communications are possible during this time) and contact duration is the interval for which a link is up.



(a) Inter-contact time distribution



(b) Contact duration distribution

Figure 4.1: Inter-contact time distribution and contact duration distribution for the ZebraNet trace. The distribution of these two metrics for four randomly selected pairs of nodes are plotted. Other links show similar characteristics. The contact duration distribution uses a different x-axis range to separate different curves.

Figure 4.1 plots the distribution of these two metrics for four randomly selected links in the ZebraNet trace. Since we observe that almost all the links in the trace show similar characteristics, we just use these four random links here as examples. As shown in Figure 4.1(a), the inter-contact time distribution has quite a few cases that a link is broken for a very long time. This observation is important because such inter-contact time patterns can lead to extremely long delays when using a naive forwarding algorithm. As expected in such a sparse network, link up-times are relatively short (as compared to the link down times) and therefore, it is important to efficiently utilize the available communication opportunity.

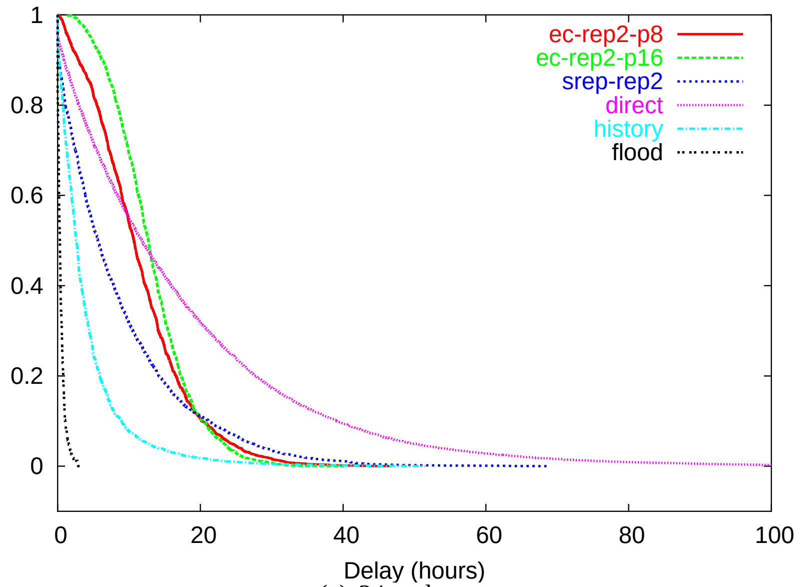
4.5.3 Impact of Node Density

Data Latency Distribution

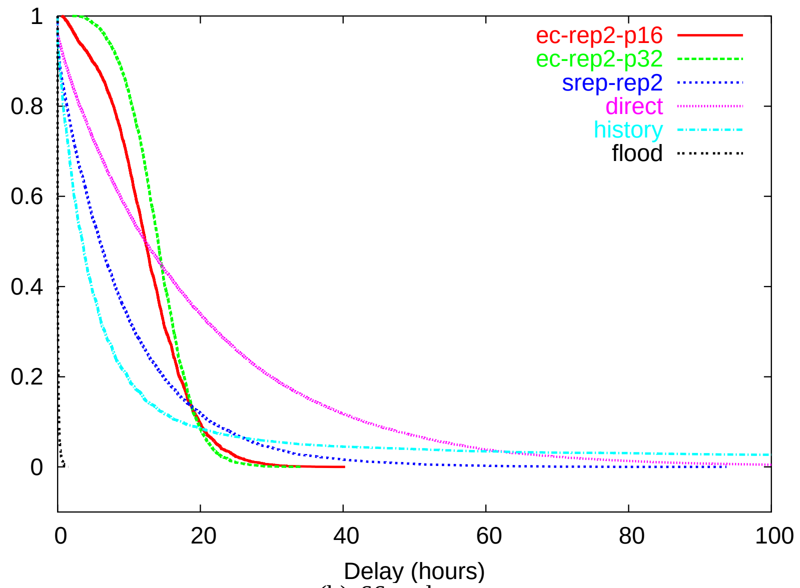
Figure 4.2 (a) and (b) shows the data latency distribution for the ZebraNet trace with 34 nodes and 66 nodes respectively. Discounting source and destination, the total number of relays are 32 and 64 respectively. The distribution is shown as a Complementary CDF (CCDF) curve.

Table 4.2 shows various data latency percentiles for both 34-node and 66-node experiments to facilitate the comparison of worse-case delay performance among all the algorithms considered.

Generally, `ec` has a higher 50^{th} percentile compared to other algorithms as shown in both Figure 4.2(a) and Figure 4.2(b) but a lower 99^{th} percentile. This is because it takes longer to find enough relays to distribute data replicas. However, once `ec` distributes enough code blocks by forwarding along multiple relays (the number of relays is larger than that used by `srep`), it takes a much shorter time to deliver the messages as it utilizes



(a) 34 nodes



(b) 66 nodes

Figure 4.2: Latency distribution for the ZebraNet trace. Traffic injection rate is 12 messages per day. The distribution is shown as a Complementary CDF (CCDF) curve. A numeric presentation of this figure is in Table 4.2 which lists the exact 50^{th}, 90^{th} and 99^{th} percentiles.

Table 4.2: Routing latency percentile (days)

Algorithm	34 nodes			66 nodes		
	50%	90%	99%	50%	90%	99%
ec-rep2-p8	0.44	0.84	1.32	—	—	—
ec-rep2-p16	0.53	0.85	1.21	0.51	0.83	1.17
ec-rep2-p32	—	—	—	0.59	0.82	1.04
srep-rep2	0.24	0.88	1.70	0.25	0.89	1.91
direct	0.49	1.63	3.27	0.51	1.79	3.54
history	0.18	0.87	9.50	0.14	0.72	10.83
flood	0.013	0.044	0.12	0.00012	0.0091	0.032

a larger number of relays and is resilient to bad performance of outlier relays. That is, in the presence of failed delivery of some of the relays, `ec` still has a good chance of message delivery via forwarding enough code blocks through other functional relays. Therefore, erasure-coding-based routing is a promising candidate for opportunistic networks where (1) relay failures are prevalent and delays are unpredictable, and (2) minimizing the worst case delay is important.

This observation is further supported by the data shown in Figure 4.2(b) with a higher node density. Given more contacts and relays, the CCDF curves of all forwarding algorithms become steeper. This is because there are more contacts overall. The `ec` approach still has the lowest 99^{th} percentile and the sharpest data latency curve. Therefore, given enough relay opportunities, `ec` delivers 99% of all messages the fastest among all algorithms considered.

Simple replication, direct contact, and history-based algorithms, on the contrary, have very long tails (messages with much longer delays). This is because they use a small number of relays and cannot guarantee if these relays will meet the destination. Very likely, some messages will encounter very long delays by selecting relays that fail to deliver the message promptly. In the long run, however, equipped with sufficient buffer space, all messages can eventually be delivered. The lower the replication factor r, the longer the tail will be. This is illustrated by comparing the CCDF of `srep-rep2` and `direct`. Since `srep-rep2` replicates its data to two other relays, the probability of losing contact opportunities is lower than that for `direct`. Hence, `srep-rep2` has a shorter tail than `direct`.

The history protocol, though having the lowest 50^{th} percentile delays, also has the longest tail among all algorithms considered. The performance of `history` is dependent on the accuracy of its selection of highest ranked relays. If the decision is fairly accurate, it tends to use relays that will deliver the data to the destination quickly. On the contrary, if the relays selected do not reflect future forwarding probabilities, very long delays may be incurred. Using certain timeout and retransmission schemes, these long-delay messages might be masked out. This will make the history protocol more attractive.

Finally, note that the `flood` protocol curves in Figure 4.2(a) and Figure 4.2(b) have latency distributions that are almost vertical. This shows that `flood` has very low delays for all messages. We use this curve as the yardstick for evaluating the performance of the other algorithms.

Routing Overhead

Table 4.3: Routing overhead

Algorithm	Overhead (34 nodes)	Overhead (66 nodes)
ec-rep2-p8	3.96	—
ec-rep2-p16	3.96	3.98
ec-rep2-p32	—	3.98
srep-rep2	3.98	3.99
direct	1.0	1.0
history	30.28	59.61
flood	68.0	132.0

Table 4.3 lists the routing overhead corresponding to each forwarding algorithm. Routing overhead is measured using the ratio of the bytes transmitted to the bytes generated. Since both `ec` and `srep` transmit a fixed amount of data with respect to the data generated, their overhead is constant. For an algorithm with a replication factor of 2, the overhead should be 4, with 2 from the source to the relay and from the relay to the destination and the other 2 for the other relay. On the other hand, in both `history` and `flood`, there are no restrictions on the replication factor and relays may also forward to other relays. This results in multiple identical copies of the original message being transmitted, depending on the selection of relays. If the relay selected has a very high history hierarchy, the overhead should be low since this node is unlikely to forward messages to other nodes with probably lower hierarchies; otherwise, the overhead may be higher than algorithms having a constant overhead. We do not use any scheme here to stop replication in our implementation of `history` and `flood`, unless the node does not meet the requirement for data forwarding in their original designs. We can reduce the overhead by timing out old messages, setting up a global budget for replication factor, or use delete list [103] to stop unnecessary replications.

As Table 4.3 shows, normally `history` has a higher overhead than `srep` and `ec`. This situation becomes worse when more contacts are available as, very likely, more duplicate messages will be transmitted to nodes that may have better delivery probabilities. For `flood`, the overhead is even larger as almost all the nodes could receive a copy before a message delivery (in that case, the overhead is proportional to $2n$, where n is the total number of nodes in the network). The factor of two comes because each relay still transmits the message to the destination in our implementation, even if it has already received a copy of the message. In summary, in terms of routing overhead, `ec` and `srep` scale well with node density and network size, while `history` and `flood` do not.

Data Success Rate

Table 4.4: Data success rate with deadlines (ZebraNet trace)

Algorithm	0.25 day	1 day	2 days	4 days	8 days
ec-rep2-p8	22.6%	95.9%	100%	100%	100%
ec-rep2-p16	9.2%	94.6%	100%	100%	100%
srep-rep2	51.8%	92.5%	99.6%	99.9%	99.9%
direct	32.0%	74.6%	94.2%	99.5%	99.9%
history	58.4%	87.9%	92.7%	94.6%	95.3%
flood	100%	100%	100%	100%	100%

Table 4.4 shows the data success rate with deadlines for different algorithms. Data success rate with deadline T is defined as the ratio of the number of messages delivered to the total number of messages generated within T. In our discussion, all deadlines are specified in units of **days**. This performance metric is used to understand delivery performance with different delay tolerance requirements. A smaller deadline indicates that the application is not tolerant to long delays and all messages delivered beyond the deadline will not be useful any more. A larger deadline, on the other hand, indicates that the application can tolerate long delays. Our results show that applications with different requirements for message deadlines should use different forwarding algorithms.

The data success rate for `ec` is low if the deadlines are less than 6 hours long. However, for relatively long deadlines (between 1 and 2 days), `ec` has the highest data success rate. This result can be observed directly by looking at the data latency distribution curve. Because `ec` has a lower 99^{th} percentile of latency distribution, it will deliver more messages before that time and hence a higher data success rate. Therefore, if achieving low latencies for all messages or high success rate within certain reasonable deadlines are of top priority to the application, `ec` should be used.

On the other hand, `history` has the highest data success rate when the deadline is less than 6 hours. This is because `history` can find good relays without the need to distribute copies of data to many relays. The performance improvement of `history` compared to `direct` and `srep` comes directly from the efficiency of its selection of good relays. However, since `history` has long tails in its data latency distribution curve, its overall data success rate is relatively low compared to other approaches.

One thing to note is partial data delivery of a message. Suppose the original message is a large one and thus has to be fragmented in the lower layers in order to be transmitted over the radio. If not all the fragments are successfully received at the sink, which we call partial delivery, we can often still retrieve useful information from the received data, using replication-based protocols. However, this may not work for coding-based protocols since a partial delivery of all the code blocks cannot be used to reconstruct the original message if the required number of blocks has not yet been received. One way to work around this is to apply erasure coding to fragmented data packets. The drawback of this approach is that it adds more coding and communication overhead because more smaller code blocks need to be processed and transmitted. This tradeoff is closely related to the implementation of the erasure-coding-based protocol and the application requirements.

4.6 Related Work

Erasure codes have been applied to many networking problems, including achieving efficient distribution of bulk data in overlay networks [14] and peer-to-peer networks [79], coping with lossy radio transmissions in wireless sensor networks [63], and achieving reliability in large-scale distributed storage systems [130]. Applying erasure coding to combat uncertainty in node mobility is the focus of this work. Since the idea of erasure-coding-based forwarding is orthogonal to the other forwarding approaches, it can potentially be combined with them to further improve routing performance and energy efficiency.

More recently, a hybrid erasure-coding-based routing algorithm [23] is proposed that aims to achieve the best of both coding-based techniques and replication-based techniques. Instead of transmitting only a fixed number of code blocks over a link, their algorithm allows the node pairs to transmit as much as possible until the link goes off. This approach works well only in scenarios where the relay nodes selected are good ones in terms of message delivery. Otherwise, it potentially misses relay opportunities that could be used to spread the data blocks to better relays. However, since no accurate knowledge of relay delivery may be available, their approach may suffer the same problem as that of purely replication-based algorithms. We conjecture that prediction-based algorithms, such as our `history` protocol, could potentially improve performance when combined with the coding-based approach. This is because such an approach will forward as much as possible only when there is a strong indication that this relay node is a good one in terms of message delivery.

Other than erasure codes, some recent efforts [60, 133] also investigate the usage of other coding techniques in DTNs and opportunistic networks. In [133], a network coding based routing algorithm is proposed that forwards packets containing information coded over the contents of several packets they received, that is, messages can be further coded at intermediate nodes. This approach will potentially further reduce the forwarding overhead as it employs new opportunities en route message transmission. Work on network coding started in [4] shows that having the routers mix information from different messages can achieve multicast capacity. The major difference between using erasure coding and network coding in challenged networks lies in that the erasure-coding-based approach only performs coding once at the source. In [60], a new code dubbed growth code is presented to maximize the amount of data that can be recovered at the sink at any point in time, even as network nodes fail. This is achieved by replicating data compactly at neighboring nodes whose complexity grow over time as data accumulates at the sink.

Prior work on simple replication forwarding that uses only one relay was shown to achieve optimal throughput in a mobile ad-hoc network [39] and has been further analyzed in [47, 54].

4.7 Conclusions and Future Work

Opportunistic networks evolve from the legacy MANETs, but encompass key features of delay/disruption tolerant networks that makes their networking protocol designs quite different from legacy MANETs and static sensor networks. For the latter, the major tasks lie in locating the best end-to-end path between communicating pairs in a resource-efficient way, which are discussed in Chapters 2 and 3. The challenges come from existing inside/outside interference, lossy links, and resource constraints. For opportunistic networks, however, the major challenge lies in supporting end-to-end communication with no end-to-end connectivity or only intermittent connectivity, which is discussed in this and the next chapter.

Prior work on routing in opportunistic networks focuses on message replication. Due to constrained energy budget in challenged networks, the performance of purely replication-based approaches are strongly dependent on their ability to select "good" relays for data forwarding, which is very difficult.

We took a different approach to this problem: using erasure coding to spread the responsibility of data forwarding over more nodes while maintaining a fixed overhead in terms of the amount of data bytes transmitted. The intuition here is to delegate the forwarding task to more nodes to avoid the impact of *outlier relays*, which refer to relays that never deliver messages as predicted. We showed via simulation on a real-world mobility trace that our coding-based approach significantly improves the worst case delay. At the same time, it has no “very small delay” cases, which is a natural consequence of the requirement to have at least r code blocks for message reconstruction.

We believe that the basic idea holds promise and an approach that combines erasure coding with prediction-based techniques, such as a prediction for link schedules, may give us good performance on both fronts. This again calls for situation awareness. For example, rather than treating all nodes as equally good relays as we did in this work, we can take advantage of any available information of the network or good predication of such information when spreading code blocks. This will improve the average delay performance because replication based approaches can achieve data delivery faster if they know which relays are better.

Chapter 5

Techniques for Improving Idle Energy

5.1 Problem and Solution Overview

Today, the study of energy efficient networking solutions for wireless sensor networks focuses on networks with always-on connectivity between communication end-points. However, maintaining end-to-end connectivity is not always possible or necessary in intermittently-connected static sensor networks. In such networks, it is very common for the network to be disconnected due to various reasons, such as environmental constraints, node failure, and intentional sleep cycles.

Engineers at Intel have studied a sensor network deployed in the North Sea aboard an operating BP oil tanker [67]. The chosen oil tanker is one of the harshest environments for industrial sensor networks. The oil tanker's aft engineering spaces are constructed of steel floors and bulkheads and are divided into three major watertight compartments with hatchways in between. Sensors are spread in the compartments to perform preventive monitoring. The hatches may be periodically shut off and as a result, the sensors within that compartment will be disconnected from the base station.

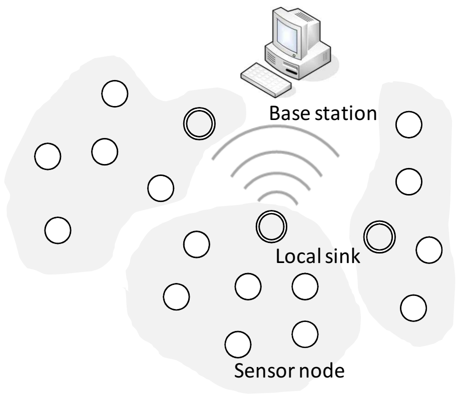


Figure 5.1: Illustration of an intermittently-connected sensor networks.

This is a typical intermittently-connected network with periodical disconnections caused by environmental constraints. Since the disconnection may last for the whole night, data generated during that period cannot be sent back to the base station immediately. Traditional sensor network protocols will continue collecting data to the local sink that will later be reconnected to the base station. However, there are two problems with such an approach. First, the local sink needs to store all data transmitted during disconnection from all the sensors connected to it, which may be a great burden to the storage of the local sink. Also, this introduces reliability problems since the local sink becomes the single failure point: its failure during disconnection will result in huge data loss. Therefore, there is no rush in pumping the data to the local sink at real time. Many other applications, such as NIMS from UCLA [7], also fall under this category.

Second, for networks with intermittent connectivity, idle energy spent on node rendezvous and idle listening during multihop routing becomes significant. Existing radios used in wireless sensor networks consume high power in their idle mode. For example, the CC2420 radio used in MicaZ and Telos motes has three modes: the **transmit/receive mode**, **idle mode**, and **sleep mode**. It consumes the lowest power in sleep mode and the most power in transmit/receive mode. When in idle mode, the radio is not communicating but the radio circuitry is still turned on. This results in a ratio of idle mode to transmit mode power (-25dBm) of 1:20, as reported in [20]. This roughly 5% energy overhead for "idle listening" becomes significant when one considers large networks where many nodes may be listening at any time. Many other wireless interfaces also show a similar trend of idle energy profile with far-from-ideal power consumption in idle mode [90].

To save energy in delay-tolerant, intermittently-connected sensor networks, data mules [108] and message ferries [146] have been widely used to exploit opportunistic or scheduled node mobility. However, in applications where no node mobility can be exploited, such as static sensor networks, we have to look at new opportunities for conserving energy, which motivates new data transport and services as described in this chapter.

In particular, we propose a new transport protocol that seeks to minimize network idle energy expenditure without compromising end-to-end data reliability. This is different from traditional transport layer solutions that often focus on reliable end-to-end delivery, flow control, and congestion control [42, 125]. Our new transport protocol — aDapTN, works by partitioning the route into regions through sleep scheduling. Therefore, only a subset of nodes along the route are awake at any specific time. By dropping the assumption of end-to-end connectivity between the source and destination, significant idle energy can be saved by placing the remaining nodes in the sleep mode.

To support such sleep scheduling, our approach requires two core components: a DTN-like store-and-forward data transport and an asynchronous wakeup scheme for node rendezvous. Store-and-forward uses custody transfer [34], a hop-by-hop reliable data transfer protocol, for data forwarding. Studies [114, 125] have shown that hop-by-hop transport protocols are more appropriate for sensor networks than end-to-end approaches, due to factors such as high link error rate. This still holds true from the perspective of energy efficiency in intermittently-connected sensor networks, as we will justify later in this chapter. We use store-and-forward transport to confine data communication to a subset of nodes, creating opportunities for nodes further along the route to sleep. An asynchronous wakeup scheme that requires no global synchronization is used for node rendezvous, when necessary. Traditional protocols, however, require all nodes along the route to be awake before multihop communication starts. Existing data transport protocols suffer from high overhead in node rendezvous [99] and can greatly benefit from the recent advances in asynchronous wakeup mechanisms [120, 147].

The store-and-forward transport and asynchronous node wakeup are both well-known techniques that deal with different networking problems. Our contribution lies in merging these two techniques to solve tough problems in harsh communication environments. We show that DTN can be applied as an energy saving technology in a constrained environment. To our knowledge, no previous work has looked at leveraging intermittent connectivity to save energy. On the contrary, other DTN variants in sensor networks, such as [72, 84], adopt store-and-forward as a means to achieve high reliability.

By exploiting these two components, however, we reduce idle energy at the expense of data latency. For different applications, users may have different requirements regarding this tradeoff. Our approach does not enforce any specific rules and exposes such controls to the applications.

To explore the relationship between traffic patterns, link delays and network diameters, we evaluate our scheme through a combination of analytical modeling and simulation. We propose analytical models for various communication models and explore their energy possibilities under different conditions. We also implement a prototype of aDapTN in TinyOS and conduct a controlled simulation study in TOSSIM [68]. Our

Model notation	**Transport**		**Rendezvous**		**Group size**	**Examples**
	intra-group	inter-group	intra-group	inter-group		
sf-async	-	store-and-forward	-	async	2	aDapTN
mh-sync	multihop	-	sync	-	h	MintRoute, AppSleep
sfk-async	multihop	store-and-forward	sync	async	k	-

Table 5.1: Classification and terminology of communication models. This table is intended to be illustrative rather than definitive. h is the total hop count from the source to the destination. k is the group size that is defined as the number of hops using synchronous rendezvous. A dash indicates that a property is unavailable to that communication model.

results show that aDapTN achieves much better idle energy efficiency than conventional approaches, without compromising data delivery rate.

5.2 Communication Models

In this section, we explore the design space of existing communication models for intermittently-connected static sensor networks, with a focus on the transport paradigm and the node rendezvous method adopted.

Transport paradigms. We classify transport paradigms into two categories: multihop transmission and store-and-forward.

In multihop transmission (**mh**), an end-to-end multihop path from the source to the sink is constructed before data transmission. A message is forwarded to the sink from a source without any delay. If a transmission to the downstream node en route fails, retransmissions are scheduled immediately to achieve reliability. The message will be dropped if it cannot be delivered after a certain number of retransmissions. We define a **group** as all nodes on the path from the source to the destination, including the source and destination. The group size is defined as the total number of nodes in a group. Intra-group communication is defined as any transmission between two nodes in the group.

In store-and-forward (**sf**), a message is stored at the intermediate nodes before it is forwarded to the next hop. If disconnection happens, the forwarding node will cache the message until a connection is restored, given that there is no storage overflow. For store-and-forward, a group consists of only the communicating two nodes at any instant and the number of groups is the same as the number of hops from the source to the destination. Inter-group communication is defined as any transmission between two nodes that belong to two different groups. If we allow a k-hop sub-network in a group, we represent it as **sfk**.

Nodes rendezvous patterns. Low-power radios usually have several power modes with different power usage profile. To save energy, power management protocols switch radios between different states while maintaining certain properties, such as the maximum data latency. Sleep scheduling, an important power management scheme, is often used in energy efficient MAC protocols, and sometimes in applications [99] to reduce idle listening time.

A basic problem introduced by the use of duty cycling as an energy saving technique is the need to establish rendezvous between the transmitter and receiver. Communication can only take place when the radios of both transmitter and receiver are active at the same time. Therefore, coordination is required between them so that their active time is overlapped. There are two types of rendezvous in general: synchronous and asynchronous.

In synchronous rendezvous, nodes in the network are time synchronized so that their active/sleep intervals happen at relatively the same time. S-MAC [140], IEEE 802.15.4 [2], and IEEE 802.11 Power Saving Mode (PSM) [1] are typical one-hop MAC protocols that use synchronous rendezvous. Multihop synchronization requires at least $n - 1$ pairwise synchronizations for n nodes, which is very expensive for a large n. AppSleep [99] takes a coarse-grain approach that synchronizes all nodes on the route periodically using a SYNCH broadcast. This has been shown to be effective for low data rate stream-oriented applications. A guard time (T_{guard}) is provided to allow for clock drifts in between SYNCHs and a radio has to be awake for $2T_{\text{guard}}$ in the worst case to guarantee pair-wise rendezvous.

Asynchronous rendezvous, on the other hand, allows individual nodes to wake up and sleep at different time without global coordination. Time synchronization is not needed to guarantee active time overlap between communicating pairs. However, this often comes at the cost of increased delay. Several asynchronous rendezvous methods [93, 120, 147] have been proposed in the literature.

5.3 Design and Energy Possibilities

In this section, we explore the design space of a range of communication models for intermittently-connected static sensor networks using sleep scheduling and present their idle energy costs via analytical modeling.

5.3.1 Assumptions

Below are the assumptions we make regarding our energy models:

- *Storage energy.* Since ultra low power storage technology is already available [78], we do not consider storage-related operations in our energy analysis.
- *Network model.* We assume that all sensors are connected to the sink in a multihop way. Nodes either are static or have only minor mobility. Since aDapTN makes few assumptions about node movement and the asynchronous wakeup scheme is agnostic to node mobility, it should also be applicable to highly mobile networks.

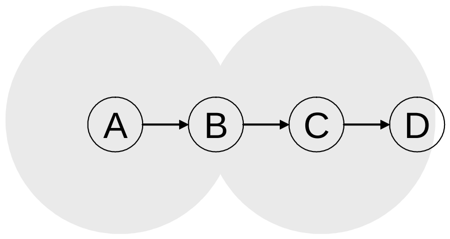


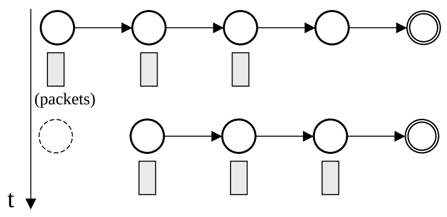
Figure 5.2: Illustration of self-interference in a multihop network.

We consider a lossless channel with no packet drops caused by unreliable links during node rendezvous and data transmission. However, self-interference between downstream and upstream traffic may result in increased backoffs that reduces the actual network throughput and increase node idle waiting time in turn. Figure 5.2 shows a four-node line topology wherein nodes A, B or C cannot transmit at the same time due to self-interference. This reduces the actual throughput to 1/3 of that with no interference. In our analysis, we assume a constant self-interference factor $r = 3$ across the whole network for transmissions in a h hop network with $h >= 3$. For single-packet messages, such as the SYNCH packets used in synchronous rendezvous, $r = 1$.

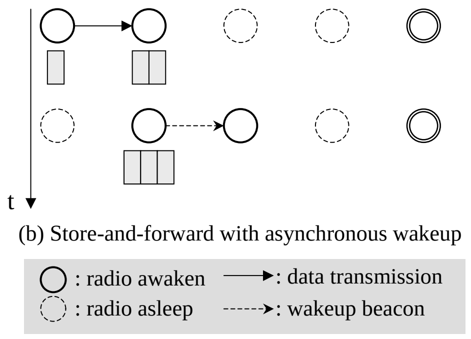
- *Application.* We consider a data collection application with periodic data communication from a subset of nodes to a sink that is connected to a backend server. The data rate is on the order of a few packets per minute.

5.3.2 Design Space

Table 5.1 shows three communication models based on the discussions in Section 5.2. The models listed here are not intended to be exhaustive but they do cover a range of designs with very different energy/latency tradeoffs. Each model is represented as T-S with T its transport paradigm and S its rendezvous method. T can be one of the following:



(a) Multihop transmission with synchronous wakeup



(b) Store-and-forward with asynchronous wakeup

Figure 5.3: Illustration of different communication models. The message is transmitted in three packets and the last-hop node is the data sink. Packets are stored at intermediate nodes for sf-async, but not for mh-async.

sf, **mh**, or **sfk**. S can be one of the following: **sync** for synchronous wakeup and **async** for asynchronous wakeup. A representative protocol for each model is presented. Figure 5.3 illustrates some of the concepts discussed here.

The first model, *sf-async*, is based on asynchronous rendezvous between any communication pair. It leverages the asynchrony inherent in the store-and-forward transport paradigm to work together with asynchronous wakeup schemes. Data transmission no longer requires all nodes to be powered on at the same time. A message is forwarded toward the sink as far as possible and is cached at the intermediate nodes where there is a disconnection, waiting for the wakeup of the next-hop node.

The second model, *mh-sync*, is used widely in conventional sensor networks. One example is MintRoute [136], which builds a collection tree based on the expected number of transmissions (ETX) to the sink. Data transmission begins only after all nodes are synchronized to be awake and ready for communication. One efficient approach of achieving such rendezvous is proposed in AppSleep [99], as described in the last section.

The third model, *sfk-async*, is different from sf-async in that it allows synchronous and asynchronous data transfer to co-exist in the network. Nodes are organized into groups with intra-group rendezvous synchronous and inter-group rendezvous asynchronous. This model can be used to take advantage of existing short link delays in a network with varying propagation delay to achieve a low data latency, while keeping the total idle energy expenditure low.

5.3.3 Idle Energy and Message Latency Analysis

In our analysis, we divide idle energy into two parts: that spent on node rendezvous (E_r^{idle}) and that spent on idle waiting during data transfer (E_d^{idle}). The total idle energy for a model m is calculated as $E_m^{\text{idle}} = E_r^{\text{idle}} + E_d^{\text{idle}}$. E_r^{idle} also consists of two parts, that spent during T_{guard} and that spent during SYNCH broadcast, respectively.

In what follows, we derive the *worst case* idle energy and message latency for these three communication models, respectively. The parameters we consider include message size, packet size, hop length, link data rate, self-interference factor, and others, with their notations listed in Table 5.2. Since our energy model is exclusively about idle energy usage, from now on we simply represent E_m^{idle} as E_m.

Store-and-forward with Asynchronous Rendezvous (sf-async)

Since each node wakes up independently using asynchronous rendezvous in sf-async, its idle waiting time is determined purely by the probability of a node stay awake, which is represented as π_w. The derivation of π_w will be deferred to Section 5.4 where the implementation of a grid quorum system is described. The total awake time is then calculated

Parameters	**Explanations**	**Units**
D	message size	bytes
n	number of packets per message	
p	packet size	bytes
h	hop count from source to destination	
b	link data rate	bits/second (bps)
r_s	self-interference factor for SYNCH packet	
r_d	self-interference factor for data transmission	
k	number of hops per group (group size)	
g	number of groups (for sfk-async)	
T_{guard}	guard time for synchronous wakeup	seconds
T_{aw}	(worst case) asynchronous wakeup delay	seconds
T_{tx}	(worst case) per-hop packet delay	seconds
π_{w}	probability of staying awake in a sleep schedule	

Table 5.2: Parameters and notation.

as $2h\pi_w T_{\mathrm{aw}}$ since each per-hop transfer requires two nodes to be awake for $\pi_w T_{\mathrm{aw}}$ in the worst case, as shown in Figure 5.3(b). Since energy is power integrated over time, the total idle energy for sf-async, therefore, is estimated as:

$$E_{\mathrm{sf-async}} = (2h\pi_w T_{\mathrm{aw}})P_{\mathrm{idle}} \tag{5.1}$$

Since node rendezvous is decoupled from data transfer in this model, any single message or a burst of messages will perform a rendezvous before data transfer, which incurs a maximum per-hop delay of T_{aw}. Also, packet transmissions are not pipelined and a message is forwarded to the next hop only when all packets of this message are completely received, which incurs another per-hop delay of $\frac{D}{p}T_{\mathrm{tx}}$. Therefore, the worst case message latency is calculated as:

$$T_{\mathrm{sf-async}} = h(T_{\mathrm{aw}} + \frac{D}{p}T_{\mathrm{tx}}) \tag{5.2}$$

Multihop Transmission with Synchronous Rendezvous (mh-sync)

Node rendezvous in mh-sync requires each node to be awake for at least $2T_{\text{guard}}$ to tolerate clock drift between SYNCH packets, as shown in Figure 5.3(a). In a multihop network, each node also needs to stay in idle mode waiting for data packets to arrive. The duration depends on the hop count from the source to this node. Thus, the total idle waiting time spent on node rendezvous in the network is $\sum_{i=1}^{h}(2T_{\text{guard}} + r_s T_{\text{tx}} i)$ with i the hop count from the source to an intermediate node on the multihop route. Similarly, $r_d T_{\text{tx}} i$ is the worst case idle waiting time for an intermediate node i hops away from the source to receive the first data packet. Therefore, the total idle energy is calculated as:

$$\begin{aligned} E_{\text{mh-sync}} &= \sum_{i=1}^{h}(2T_{\text{guard}} + (r_s + r_d)T_{\text{tx}} i)P_{\text{idle}} \\ &= \big(2hT_{\text{guard}} + \frac{1}{2}(r_s + r_d)h(h+1)T_{\text{tx}}\big)P_{\text{idle}} \end{aligned} \tag{5.3}$$

In this model, node rendezvous is coupled with data transfer and is done only once before data transfer, which takes hT_{tx} to complete. Packet transmissions occur in a pipelined fashion which delivers one packet every T_{tx} once the pipeline is full. However, the pipeline is not full until the first packet reaches the destination h hops away, adding a latency of hT_{tx}. As a result, the worst case latency of the message is $(n-1+h)T_{\text{tx}}$. Since data packets are only transmitted after node rendezvous is done, we do not include the latency of node rendezvous as part of message latency and calculate it as:

$$T_{\text{mh-sync}} = r_d(n-1+h)T_{\text{tx}} \tag{5.4}$$

A Hybrid Model (sf$k$-async)

In this model, nodes are formed into groups: rendezvous between groups is asynchronous while that inside a group is synchronous. Since inter-group rendezvous only needs the two edge nodes from each group to be involved, the idle waiting time spent on inter-group rendezvous is only proportional to the number of groups g. Let k be the number of hops per group. Then we have $kg + (g - 1) = h$ with kg the total number of hops within groups and $g - 1$ the number of hops in between the g groups. Therefore, g is calculated as $\lceil \frac{h+1}{k+1} \rceil$.

The idle waiting time spent on inter-group communications can be calculated as $2g\pi_w T_{\text{aw}}$ by simply replacing h with g in Eq. (5.1). Similarly, the per-group idle waiting time spent on intra-group communication can be calculated by replacing h with k in Eq. (5.3) as $2(k+1)T_{\text{guard}} + \frac{1}{2}(r_s + r_d)k(k+1)T_{\text{tx}}$. The total idle energy is then calculated as:

$$\begin{aligned} E_{\text{sfk}-\text{async}} &= \big(2g\pi_w T_{\text{aw}} + g(2(k+1)T_{\text{guard}} + \\ &\qquad \frac{1}{2}(r_s + r_d)k(k+1)T_{\text{tx}})\big)P_{\text{idle}} \end{aligned} \tag{5.5}$$

The latency of transmissions within a group is calculated by replacing h with k in Eq. (5.4) as $r_d(n - 1 + k)T_{\text{tx}}$. The latency of inter-group communication can be treated as a sf-async model with $g - 1$ virtual hops, which is calculated as $(g - 1)(T_{\text{aw}} + \frac{D}{p}T_{\text{tx}})$ by replacing h with $g - 1$ in Eq. (5.2). Therefore, the total message latency for this model is calculated as:

$$T_{\text{sfk}-\text{async}} = gr_d(n - 1 + k)T_{\text{tx}} + (g - 1)(T_{\text{aw}} + \frac{D}{p}T_{\text{tx}}) \tag{5.6}$$

	CC2420
Transmit	28.1mW (-25dBM)
Receive	62.1mW
Idle power (P_{idle})	1.41mW

Table 5.3: Reported power numbers of CC2420.

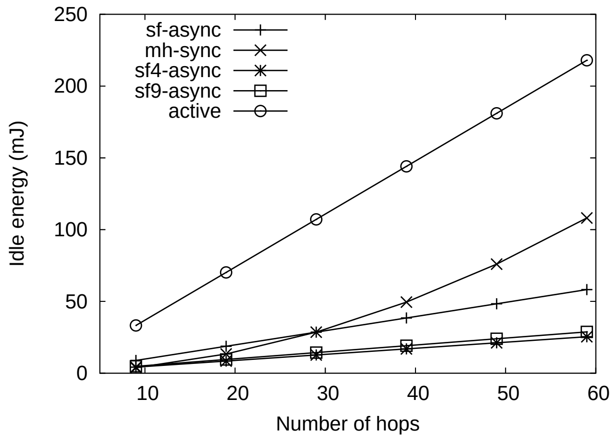


Figure 5.4: Idle energy expenditure. There are a total of 5 groups for sfk-async.

5.3.4 Analytical Results

The energy related parameters used for our analysis is based on the CC2420 family of low-power, 802.15.4-compatible radios from Chipcon, which have been used in many sensor platforms. We use the published data from [20], as listed in Table 5.3.

Figure 5.4 plots the worst case idle energy expenditure, with a message size of 512 bytes and a link rate of 100Kbps. Figure 5.5 plots the worse case message latency. We use a message transmission delay of 0.01s that is typical for wireless sensor networks [93]. Since node rendezvous is conducted only once for mh-sync, we conjecture that aDapTN will gain more benefits for small messages since the rendezvous cost is amortized across all packets in mh-sync. Active energy refers to that spent on radio transmissions and is plotted here to show the relative importance of idle energy expenditure. We assume loss-less channels and no transmission contention. In reality, packets may be retransmitted,

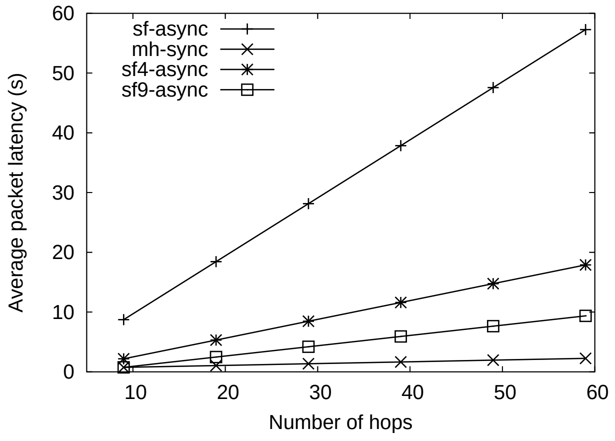


Figure 5.5: Data latency. We use two group sizes (4 and 9) for sfk-async.

which produces higher active energy expenditure. However, the relative trends shown in the figures will still hold as retransmissions normally will not change the order of magnitude of energy consumption.

Figure 5.4 shows that active energy dominates idle energy. However, as hop counts increase, idle energy expenditure grows much faster since the total idle energy cost grows exponentially with increased hop counts. Therefore, for networks with very long routes, sfk-async will gain more benefits. Also, sf-async, sf4-async and sf9-async all outperform mh-sync because they do not require all nodes en route to be in idle mode. Compared to sf9-async, sf4-async has a lower idle energy expenditure. This is because more nodes are participating asynchronous communications in sf4-async than in sf9-async. The same reasoning can be used to explain the difference between sf-async and sfk-async (k=4,9).

Figure 5.5 shows that mh-sync has the lowest data latency among all models considered. Comparing sf-async and sfk-async (k=4,9), we see that sfk-async achieves significant improvement in data latency and a slightly higher idle energy usage. This comes as no surprise since sfk-async has a group size of k and thus fewer number of inter-group hops, which is proportional to the worse case data latency. Further, since the idle energy expenditure for mh-sync grows exponentially with the number of hops, the idle energy increases only slightly for small group sizes. Hence, sfk-async ($k > 2$) achieves a good balance between idle energy expenditure and data latency. We can, therefore, adjust the group size (k) to meet different application preferences regarding the energy/latency tradeoff.

5.4 Design and Implementation

In this section, we describe the architecture of aDapTN and provide details about our prototype implementation.

5.4.1 Core Algorithms

As a concrete implementation of the generic communication model described in previous sections, aDapTN consists of core algorithms related to node rendezvous, transport, etc., which are explained in this section.

Quorum-based Wakeup

We use a quorum-based wakeup scheme, as proposed in [120], to wake up the next-hop node in a multi-hop network. The quorum system we used is a grid quorum system with applications in many other areas, such as distributed mutual exclusion [75]. In brief, if we divide one round of schedule into q^2 time slots, the radio only needs to be powered on for $2q - 1$ of the schedule duration to guarantee rendezvous with another node to ensure one communication. The selection of q is a design parameter. A higher q will result in very efficient power usage. However, it will lead to longer delay. An example grid quorum system with $q = 4$ is illustrated in Figure 5.6. Each grid represents the quorum system

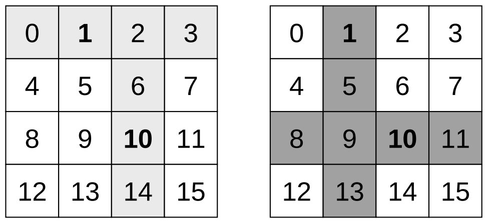


Figure 5.6: An example of a grid quorum system with $n = 4$. The two quorum groups overlap at time slots 1 and 10.

used by one node. We call the time slots that a node needs to be awake a *quorum group* and the length of each such time slot a *quorum interval*. The radio should be either on or off during each slot and it only needs to wake up in the quorum group. The two nodes use different quorum groups shown as the shaded region in the matrix. The highlighted regions are those in which the two nodes overlap.

In each quorum interval, the node needs to send out a beacon message first for synchronizing with other potential neighbors, as illustrated in Figure 5.7. Once two nodes are synchronized with each other, they can keep on communicating until all buffered messages are transmitted. Then, they can resume their normal schedules independently again and wait for the next rendezvous.

If we assume the clocks of the two nodes are synchronized, it is easy to see that such a quorum system will overlap twice for one round of schedule. However, we can prove that even if their clocks are not synchronized, they are guaranteed to hear each other's beacon message at least once for each round using our wakeup method.

Our transport protocol can be used with any MAC layer protocol that handles the micro-level issues such as channel contention, hidden terminal problems, etc. The data collection schedule is controlled by aDapTN separately at a macro-level. This approach keeps the MAC layer simple and allows for reuse of well-understood MAC protocols.

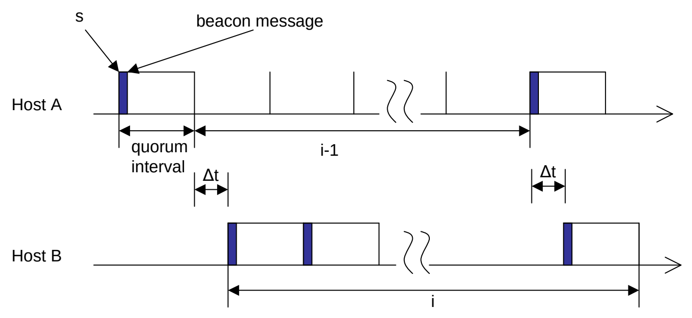


Figure 5.7: An example of the quorum-based asynchronous wakeup.

Tree Construction

For each rendezvous, aDapTN exchanges route update information and link quality estimation information as in MintRoute. Though the dissemination of routing information may be slow compared to MintRoute, our protocol can still find a good end-to-end path if adjacent node pairs can talk to each other within each round of schedule.

Rendezvous Contention

Since the quorum group selected by each node is randomly initialized, two or more nodes could be competing to be synchronized with a node that is awake. This introduces contentions during rendezvous. Even worse, beacon messages used for node rendezvous may interfere with normal traffic transmissions. To address this problem, we assign different priorities to different traffic. For those that require immediate response, we assign a high priority by setting their expiration timers to the smallest in order for them to grab the channel first. In aDapTN, priority is assigned in decreasing order to the following types of traffic: data transmission traffic, routing update traffic and other control message exchanges. This simple scheme works very well for reducing contention and interference among different traffic.

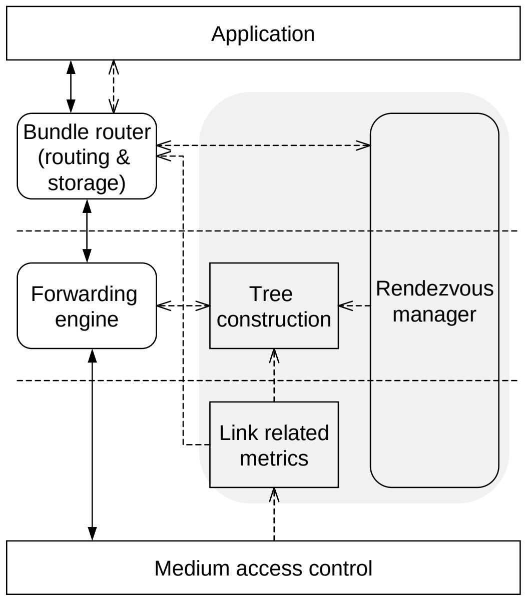


Figure 5.8: Component block diagram of aDapTN.

5.4.2 System Architecture

This section describes our prototype implementation of aDapTN in TinyOS, with its system structure shown in Figure 5.8. The shaded blocks comprise the control plane, which includes the routing stack and the rendezvous manager that controls the synchronization between communicating nodes. In our current design, we choose a tree-based data collection routing protocol similar to MintRoute. Other networking protocols, such as geographic routing, can also be used here. The rest of the components work together to forward data messages. Currently, our implementation of aDapTN supports sf-async.

5.5 Evaluation

To better understand aDapTN's behavior and design tradeoffs, we evaluate it using simulation in this section. The simulator models the wireless channel behavior based on packet loss distribution data collected from a real-world testbed. Although not perfect, it allows us to quickly examine the performance of aDapTN.

5.5.1 Experiment Setups

We use the TOSSIM-CC2420 simulator[1] provided in the TinyOS distribution [117]. It models the CC2420 radio with a data rate close to 100kbps. This matches the environment of the BP application described in Section 5.1. TOSSIM-CC2420 incorporates PowerTOSSIM [109], a power modeling extension to TOSSIM. PowerTOSSIM can model power consumed by TinyOS applications on Mica2, MicaZ, and Telos motes.

We compare the performance of aDapTN with MintRoute, which needs to first wake up all nodes involved in the communication and then pumps data towards the sink in a multihop fashion. Since the original design of MintRoute does not specify ways to achieve node rendezvous, we use the scheme proposed in AppSleep for this purpose and use MintRoute as the multihop transmission scheme.

We use a low-rate data collection application for our evaluations. The message size is 30 bytes that can easily fit into one packet in TinyOS. The message arrival rate is set at 4 pkts/min. The quorum size is set at 16 and the quorum interval is set at 1200ms. We use two topologies, a 3x3 grid topology and a 1x12 line topology, to create networks with different hop counts. For the 3x3 topology, node 9 is selected as the source and node 1 as the sink; for the 1x12 topology, node 12 is selected as the source and node 1 as the sink. Each experiment lasts 600s. We run each experiment 5 times and the average is shown.

5.5.2 Performance Metrics

We consider the following performance metrics in our evaluation.

[1]`tinyos-1.x/beta/TOSSIM-CC2420/`

- *Success rate:* The fraction of messages that are delivered to the sink. Since data reliability is of high priority in many applications, our proposed scheme should have at least a success rate as high as other conventional approaches. In our experiments, since both MintRoute and aDapTN achieve the same success rate for the scenarios we simulated, we omit the success rate comparison.

- *Average idle energy per node:* The idle energy expenditure averaged among all nodes.

- *Average data latency:* The average message latency among all successfully delivered messages.

5.5.3 Results

Table 5.4 shows the idle energy results for both MintRoute and aDapTN. They demonstrate that for both the 3x3 and 1x12 topologies, aDapTN spends much less time in idle mode to deliver the same amount of data. It achieves idle energy savings at the cost of increased latency. For the 3x3 topology, the average latency is 75s. For the 1x12 line topology, the average latency is 228s due to increased hop counts from the source to the sink. However, the delays are still within several minutes. For a typical data collection application, this is acceptable, given that almost half of the idle energy can be saved compared to conventional approaches.

For networks with very low-rate links and long propagation delays, we postulate that aDapTN can gain even more benefit in terms of idle energy savings because idle waiting time saved is proportional to link delays and inversely proportional to data rates.

Furthermore, if a node and its neighbors can only discover each other in an asynchronous way, the relay nodes selected may not be the optimal ones. Given that (1) we

	3x3 grid		1x12 line	
Communication model	**Energy per node (mJ)**	**Average latency (s)**	**Energy per node (mJ)**	**Average latency (s)**
mh-sync (MintRoute, AppSleep)	6616	7	6683	16
sf-async (aDapTN)	3264	75	3358	228

Table 5.4: Experimental performance using two different topologies.

Quorum size	Energy per node (mJ)	Average latency (s)
16	3264	75
36	2481	774

Table 5.5: Performance of aDapTN with different quorum size (16 vs. 36).

can control the delay of control information exchanges using asynchronous wakeup, and (2) link quality usually will not fluctuate sharply during small time intervals, we can still have a routing structure that is close to the optimal one.

Impact of Quorum Size

As we introduced in Section 5.4, a grid-quorum system can tune its parameter q to trade in energy with data latency or vice versa. For the above experiments, we use a quorum group of 16 ($q = 4$) time slots. Therefore, during each round of schedule, the radio needs to be on for 7 time slots. For this experiment, we change the quorum group size to 36 ($q = 6$). Hence, each node needs to be on 11 of the 36 time slots. Analytically, this will produce idle energy savings of $(7/16 - 11/36)/(7/16) = 30.2\%$.

Table 5.5 shows the tradeoffs using different quorum size. For the same amount of running time, using a quorum size of 36 will produce energy savings of 24.0%. This is close to the analytically estimated number. Many factors can contribute to the slight difference, such as packet retransmissions, which are not considered in the analytical model.

5.6 Related Work

Our work is related to delay/disruption tolerant networks, asynchronous wakeup in mobile ad hoc networks, and power management in sensor networks. We discuss the most relevant work in each category as follows.

Delay/Disruption Tolerant Networks (DTNs). The DTN architecture presented in [33] provides a generic network architecture for various challenged networks. The authors in [43] discuss various ways to apply the DTN architecture to sensor networks. DTNlite [84] presents a real implementation of a stripped-down version of the DTN architecture in TinyOS on resource-constrained motes. However, their approach assumes an always-on network that is not suitable for more challenged sensor networks. The only work we are aware of on DTN power management is [59]. Their approach targets mobile networks and their goal is on maximizing contact opportunities between nodes when such power management is used. Furthermore, their approach assumes that nodes are time-synchronized which is a strong assumption in challenged sensor networks. Our approach, however, can work even if node clocks are not synchronized.

Asynchronous wakeup. Tseng *et al.* [120] propose three asynchronous power management protocols for mobile ad hoc networks where synchronized power management is difficult, such as networks with unpredictable node mobility and networks with no clock synchronization mechanism. Later on, they identify in [55] a rotation closure property that allows for a more flexible quorum system design. Zheng *et al.* [147] propose an asynchronous wakeup scheme based on block combinatorics design and an on-demand power management protocol based on it.

Power management. A stream-oriented power management protocol is proposed in [99] to support a class of sensor network applications characterized by delay tolerant, asynchronous data traffic, and scheduled data transmission. An application-layer wakeup/sleep scheme is proposed to enable energy-efficient network operations by only keeping the active route between a source and receiver awake. This scheme relies on the existence of a stable and fixed end-to-end route during the entire data stream transmission and does not apply to many challenged networks.

5.7 Conclusions and Future Work

In this chapter, we have presented aDapTN, a new transport protocol that saves significant idle energy in delay-tolerant, static sensor networks. Our technique consists of two core components: a store-and-forward transport and an asynchronous node rendezvous. It saves idle energy by relaxing the requirement for end-to-end connectivity during data transmission and allowing the network to be disconnected intermittently via scheduled sleeping. To our knowledge, no previous work has leveraged intermittent connectivity to save energy in such environments. Beyond the applications we discussed, we expect our approach to be useful to other challenged networks where idle energy efficiency is crucial.

Due to the limitations of our experiment environments, we have not evaluated the feasibility of aDapTN in settings with disconnections caused by other reasons, such as high packet loss rates and node failures. We expect that aDapTN will perform well in such environments due to its robustness to any type of disconnection and we leave it as future work. Another avenue for future work is an adaptive transport protocol that can dynamically switch between different communication models when the environment changes. The group-oriented model is one such protocol. Efficient design of such a protocol is an area of future work.

Overall, this work offers significant opportunities for saving idle energy, and further work has high potential for improving it.

Chapter 6

Conclusions and Future Work

Communications in emerging challenged networks present new complexities, such as intermittent connectivity, and extremely unreliable radio transmissions. The work in this dissertation explores techniques to provide and leverage situation information as a means to improve routing efficiency and performance. It has brought a unique approach to data communication performance optimizations in such networks.

To elaborate the main ideas of this research, this dissertation investigates a range of challenges that cover several key areas of challenged networks, including wireless mobile and static sensor networks, opportunistic networks, and DTNs. The first part of this dissertation focuses on understanding and exposing such abnormal situations, such as the varying link quality observed in many low-power radio transceivers. The second part focuses on novel routing design in networks confronted with challenging disruptions with no well-established solutions, such as those found in opportunistic networks.

This dissertation demonstrates that many lower-layer system and network metrics that are readily available can effectively help improve routing performance in challenged networks. One important thing to note is that such metrics should be easy to collect with no or low overhead, due to cost and other practical concerns. To accommodate extreme conditions in challenged networks, it is important for the protocols to make no explicit assumptions of the network and to treat disruptions as first-order design parameters.

By designing, building and evaluating our solutions on real-world sensor network testbeds serving real traffic, or using real-world mobility traces, we successfully factor real-world issues into our evaluations. Our model-based protocol can capture mobility phase changes with high accuracy and achieves an improvement of up to 120% in packet delivery rate (Chapter 2). Our supervised-learning based technique can maintain accurate link quality information even under heavy traffic load when traditional approaches fail. Used with link-quality aware routing protocols, it can yield performance improvements of up to 300% in packet delivery rate (Chapter 3). Our erasure-coding based forwarding protocol has a consistently smaller worst case message delay than four other state-of-the-art forwarding algorithms, when evaluated in an opportunistic network with intermittent connectivity and unpredictable mobility (Chapter 4). Our new transport protocol, aDapTN, when used in networks with intermittent connectivity, can yield significant idle energy savings compared to existing approaches (Chapter 5).

We believe that our findings are general enough to be useful to other systems and networks. In essence, techniques proposed for one type of challenge can often apply to other challenges. However, sometimes we need more care in applying such techniques to different environments. For example, in an opportunistic network with unpredictable node mobility, it becomes difficult to apply the centralized supervised-learning technique since important samples of interest may not be available.

In the long term, researchers are envisioning a global network with devices of all types and capabilities being connected and integrated seamlessly. To accomplish such a goal, future networking solutions are anticipated to have intelligence embedded to understand their operating environment and make informed decisions. Such a communication paradigm shift from networks nowadays to large-scale heterogeneous networks renders our research even more relevant and the focal points of this dissertation will play an important role in networked applications of tomorrow.

6.1 Future Research

Challenged wireless and mobile networks, in particular wireless sensor networks, have evolved a long way in the last few years. Other than environmental sensing scientific applications, new sensing applications are on the rise that encompass human involvement and mobility and open a new range of problems to be explored. We have made significant progress in understanding and developing solutions for efficient communication in challenged networks, including wireless sensor networks, mobile networks, and their hybrids. We find that the cross-fertilization of the two fields poses new challenges as well as opportunities, as human mobility plays an increasingly important role. Such networks are referred to as *participatory* networks where participants are central to the network functioning, compared to the first generation sensor networks whose major goal is confined to data collection. Yet, substantial work remains to achieve a comprehensive solution to communication problems in such networks. In what follows, we discuss several directions for future work that merit further investigation.

6.1.1 Hybrid Sensing Applications

The existence of mobility introduces a new set of tradeoffs between energy and response time, and opens possibilities of new optimizations in cost. To take advantage of mobility, a store-carry-and-forward routing paradigm is usually exploited. This requires the nodes to be able to efficiently manage a large amount of data locally. Compared to the evolution of processors, radios, and batteries, non-volatile memory based storage is improving greatly in cost and energy efficiency over the years and the limitations in hardware should not be a concern. Rather, what remains a concern is that such a storage-centric communication pattern may lead to reliability, energy and performance problems.

6.1.2 Heterogeneity

Heterogeneity is another distinct characteristic of participatory sensor networks. A heterogeneous network encompasses a diverse set of wireless devices with different computation, communication, storage, and mobility capabilities. For example, embedded sensing devices in the near future can have multiple radios [83], cognitive radios [85, 105], or renewable energy supply [44, 61]. Efficiently leveraging the power of such heterogeneous devices requires the networking protocol to identify the tradeoffs with new hardware in order to fully take advantage of their new capabilities.

6.1.3 Data Quality

Previous energy-efficient designs are focused on maximizing a certain pre-defined goal, such as the system lifetime or data yield. Therefore, they do not accommodate different requirements for the collected data. In the real world, however, users have very different requirements regarding data output, such as sensor readings or location-related information. For such networks, a knob to control the tradeoff between data yield and energy is attractive, compared to a static system that only supports one quality goal or another. Such a quality-control property is particularly useful in participatory sensing networks because dynamic human engagement will definitely lead to different quality requirements.

In conclusion, this dissertation has shown that exposing and leveraging situation information can yield significant performance gains and provides a set of such techniques. The outcomes reveal the potential of situation-aware techniques and provide new perspectives on performance optimizations in challenged networks.

Bibliography

[1] *IEEE Standard 802.11: Wireless LAN Medium Access Control and Physical Layer Specifications*. 1999.

[2] *IEEE Standard 802, part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)*. 2003.

[3] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *Proceedings of the International Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2004.

[4] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4), July 2000.

[5] F. Bai, N. Sadagopan, and A. Helmy. IMPORTANT: A framework to systematically analyze the Impact of Mobility on Performance of RouTing protocols for Adhoc NeTworks. In *Proceedings of the IEEE INFOCOM*, 2003.

[6] S. Bapat, V. Kulathumani, and A. Arora. Analyzing the yield of exscal, a large-scale wireless sensor network experiment. In *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, 2005.

[7] M. A. Batalin, M. Rahimi, Y. Yu, D. Liu, A. Kansal, G. S. Sukhatme, W. J. Kaiser, M. Hansen, G. J. Pottie, M. Srivastava, and D. Estrin. Call and response: experiments in sampling the environment. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[8] B. Bougard, F. Catthoor, D. C. Daly, A. Chandrakasan, and W. Dehaene. Energy efficiency of the IEEE 802.15.4 standard in dense wireless microsensor networks: Modeling and improvement perspectives. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, 2005.

[9] J. A. Boyan and M. L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in Neural Information Processing Systems,* volume 6. Morgan Kaufmann Publishers, 1994.

[10] E. Brewer, M. Demmer, B. Du, K. Fall, M. Ho, M. Kam, S. Nedevschi, J. Pal, R. Patra, and S. Surana. The case for technology for developing regions. *IEEE Computer*, 38(6), June 2005.

[11] E. Brewer, M. Demmer, M. Ho, R. Honicky, J. Pal, M. Plauch, and S. Surana. The challenges of technology research for developing regions. *IEEE Pervasive Computing*, 5(2), April-June 2006.

[12] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 1998.

[13] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. In *Proceedings of the workshop on World Sensor Web (WSW)*, 2006.

[14] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost. Informed content delivery across adaptive overlay networks. In *Proceedings of the International Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2002.

[15] B. Calder, D. Grunwald, D. Lindsay, M. Jones, J. Martin, M. Mozer, and B. Zorn. Evidence-based static branch prediction using machine learning. *ACM Transactions on Programming Languages and Systems*, January 1997.

[16] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. Peterson. People-centric urban sensing. In *Proceedings of the ACM/IEEE Annual International Wireless Internet Conference (WICON)*, 2006.

[17] Q. Cao, T. He, L. Fang, T. Abdelzaher, J. Stankovic, and S. Son. Efficiency centric communication model for wireless sensor networks. In *Proceedings of the IEEE INFOCOM*, 2006.

[18] M. Carvalho and J. Garcia-Luna-Aceves. Delay analysis of IEEE 802.11 in single-hop networks. In *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, 2003.

[19] M. M. Carvalho and J. Garcia-Luna-Aceves. A scalable model for channel access protocols in multihop ad hoc networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2004.

[20] Chipcon CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver. http://www.chipcon.com/.

[21] A. Cerpa, J. Wong, M. Potkonjak, and D. Estrin. Temporal properties of low-power wireless links: Modeling and implications on multi-hop routing. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2005.

[22] A. Cerpa, J. L. Wong, L. Kuang, M. Potkonjak, and D. Estrin. Statistical model of lossy links in wireless sensor networks. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, 2005.

[23] L.-J. Chen, C.-H. Yu, T. Sun, Y.-C. Chen, and H. hua Chu. A hybrid routing approach for opportunistic networks. In *Proceeding of the ACM SIGCOMM Workshop on Challenged Networks (CHANTS)*, 2006.

[24] M. Chen, A. X. Zheng, J. Lloyd, M. I. Jordan, and E. Brewer. Failure diagnosis using decision trees. In *Proceedings of the IEEE International Conference on Autonomic Computing (ICAC)*, 2004.

[25] I. Cohen, M. Goldszmidt, T. Kelly, J. Symons, and J. Chase. Correlating instrumentation data to system states: A building block for automated diagnosis and control. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2004.

[26] W. W. Cohen. Fast effective rule induction. In *Proceedings of the International Conference on Machine Learning (ICML)*, 1995.

[27] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2003.

[28] Delay-tolerant networking architecture. http://www.ietf.org/rfc/rfc4838.txt.

[29] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2000.

[30] Delay tolerant networking. http://www.dtnrg.org/.

[31] D. Estrin, D. Culler, K. Pister, and G. Sukhatme. Connecting the physical world with pervasive networks. *IEEE Pervasive Computing*, 1(1):59–69, 2002.

[32] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: scalable coordination in sensor networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 1999.

[33] K. Fall. A delay-tolerant network architecture for challenged Internets. In *Proceedings of the International Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2003.

[34] K. Fall, W. Hong, and S. Madden. Custody transfer for reliable delivery in delay tolerant networks. Technical Report IRB-TR-03-030, Intel Research Berkeley, July 2003.

[35] Fidonet. http://en.wikipedia.org/wiki/FidoNet.

[36] R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, and S. S. I. Stoica. Beacon Vector Routing: Scalable point-to-point routing in wireless sensornets. In *Proceedings of USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2005.

[37] R. K. Ganti, P. Jayachandran, H. Luo, and T. F. Abdelzaher. Datalink streaming in wireless sensor networks. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.

[38] O. Gnawali, B. Greenstein, K.-Y. Jang, A. Joki, J. Paek, M. Vieira, D. Estrin, R. Govindan, and E. Kohler. The tenet architecture for tiered sensor networks. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.

[39] M. Grossglauser and D. Tse. Mobility increases the capacity of ad-hoc wireless networks. *IEEE/ACM Transactions on Networking*, 10(4), Aug 2002.

[40] C. Guestrin, P. B. adn Romain Thibaux, M. Paskin, and S. Madden. Distributed regression: an efficient framework for modeling sensor network data. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, 2004.

[41] T. He, B. M. Blum, J. A. Stankovic, and T. F. Abdelzaher. AIDA: Adaptive application independent data aggregation in wireless sensor networks. *ACM Transactions on Embedded Computing System Special Issue on Dynamically Adaptable Embedded Systems*, May 2004.

[42] S. Heimlicher, R. Baumann, M. May, and B. Plattner. The transport layer revisited. In *Proceedings of the IEEE International Conference on Communication System Software and Middleware (COMSWARE)*, 2007.

[43] M. Ho and K. Fall. Poster: Delay tolerant networking for sensor networks. In *IEEE SECON*, 2004.

[44] J. Hsu, S. Zahedi, A. Kansal, M. Srivastava, and V. Raghunathan. Adaptive duty cycling for energy harvesting systems. In *Proceedings of the international symposium on Low power electronics and design (ISLPED)*, 2006.

[45] Y.-C. Hu and D. B. Johnsoan. Caching strategies in on-demand routing protocols for wireless ad hoc networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2000.

[46] Y.-C. Hu and D. B. Johnson. Exploiting MAC layer information in higher layer protocols in multihop wireless ad hoc networks. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, 2004.

[47] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Pocket switched networks and human mobility in conference environments. In *Proceeding of the ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN)*, 2005.

[48] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. CarTel: a distributed mobile sensor computing system. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.

[49] B. Hull, K. Jamieson, and H. Balakrishnan. Mitigating congestion in wireless sensor networks. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[50] M. Ilyas. *The Handbook of Ad hoc Wireless Networks*. CRC Press, 2002.

[51] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2000.

[52] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling*. John Wiley & Sons, Inc., 1991.

[53] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *Proceedings of the International Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2004.

[54] S. Jain, R. C. Shah, G. Borriello, W. Brunette, and S. Roy. Exploiting mobility for energy efficient data collection in sensor networks. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, 2004.

[55] J.-R. Jiang, Y.-C. Tseng, C.-S. Hsu, and T.-H. Lai. Quorum-based asynchronous power-saving protocols for IEEE 802.11 ad hoc networks. *Mobile Networks and Applications*, 10(1-2), 2005.

[56] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[57] E. P. C. Jones, L. Li, and P. A. S. Ward. Practical routing in delay-tolerant networks. In *Proceeding of the ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN)*, 2005.

[58] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. In *Proceedings of the international conference on Architectural support for programming languages and operating systems (ASPLOS)*, 2002.

[59] H. Jun, M. H. Ammar, and E. W. Zegura. Power management in delay tolerant networks: A framework and knowledge-based mechanisms. In *Proceedings of the IEEE SECON*, 2005.

[60] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein. Growth codes: maximizing sensor network data persistence. *Proceedings of the International Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2006.

[61] A. Kansal, J. Hsu, M. Srivastava, and V. Raghunathan. Harvesting aware power management for sensor networks. In *Proceedings of the Annual Conference on Design Automation (DAC)*, 2006.

[62] M. Kim, D. Kotz, and S. Kim. Extracting a mobility model from real user traces. In *Proceedings of the IEEE INFOCOM*, 2006.

[63] S. Kim, R. Fonseca, and D. Culler. Reliable transfer on wireless sensor networks. In *Proceedings of the IEEE SECON*, 2004.

[64] C. E. Koksal and H. Balakrishnan. Quality-aware routing metrics for time-varying wireless mesh networks. *IEEE Journal on Selected Areas of Communication Special Issue on Multi-hop Wireless Mesh Networks*, 24(11), November 2006.

[65] D. Kotz and T. Henderson. CRAWDAD: A community resource for archiving wireless data at dartmouth. *IEEE Pervasive Computing*, 4(4), 2005.

[66] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: maximizing information while minimizing communication cost. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, 2006.

[67] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, and M. Yarvis. Design and deployment of industrial sensor networks: Experiences from the north sea and a semiconductor plant. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[68] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[69] B. Liblit, A. Aiken, A. X. Zheng, and M. I. Jordan. Bug isolation via remote program sampling. In *Proceedings of the ACM Conference on Programming Language Design and Implementation (PLDI)*, 2003.

[70] A. Lindgren, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. In *Proceedings of the International Workshop on Service Assurance with Partial and Intermittent Resources (SAPIR)*, 2004.

[71] T. Liu, C. Sadler, P. Zhang, and M. Martonosi. Implementing software on resource-constrained mobile sensors: Experiences with impala and ZebraNet. In *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2004.

[72] M. Loubser. Delay tolerant networking for sensor networks. Technical Report T2006:01, Swedish Institute of Computer Science, January 2006.

[73] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann. Practical loss-resilient codes. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, 1997.

[74] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A tiny aggregation service for ad-hoc sensor networks. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2002.

[75] M. Maekawa. A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems. *ACM Transactions on Computer Systems (TOCS)*, 3(2), 1985.

[76] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the ACM international workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.

[77] M. Marina and S. Das. Performance of route caching strategies in dynamic source routing. In *Proceedings of the International Workshop on Wireless Networks and Mobile Computing (WNMC) in conjunction with International Conference on Distributed Computing Systems (ICDCS)*, 2001.

[78] G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy. Ultra-low power storage for sensor networks. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN), Special Track on Platform Tools and Design Methods for Network Embedded Sensors (SPOTS)*, 2006.

[79] P. Maymounkov and D. Mazieres. Rateless codes and big downloads. In *Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.

[80] Mistlab. http://mistlab.csail.mit.edu/.

[81] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[82] M. Mitzenmacher. Digital fountains: A survey and look forward. In *IEEE Information Theory Workshop (ITW)*, 2004.

[83] A. K. Miu, H. Balakrishnan, and C. E. Koksal. Improving loss resilience with multi-radio diversity in wireless networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2005.

[84] S. Nedevschi and R. Patra. DTNLite: A reliable data transfer architecture for sensor networking. In *Proceedings of the IEEE International Conference on Intelligent Engineering Systems (INES)*, September 2004.

[85] M. Neufeld, J. Fifield, C. Doerr, A. Sheth, and D. Grunwald. SoftMAC – flexible wireless research platform. In *Proceedings of the Workshop on Hot Topics in Network (HotNets)*, 2005.

[86] H. X. Nguyen and P. Thiran. Using end-to-end data to infer lossy links in sensor networks. In *Proceedings of the IEEE INFOCOM*, 2006.

[87] V. N. Padmanabhan, L. Qiu, and H. J. Wang. Passive network tomography using bayesian inference. In *Proceedings of the ACM SIGCOMM Workshop on Internet measurment*, 2002.

[88] J. Paek, K. Chintalapudi, R. Govindan, J. Caffrey, and S. Masri. A wireless sensor network for structural health monitoring: Performance and experience. In *Proceedings of the Workshop on Embedded Networked Sensors (EmNets)*, 2005.

[89] L. Pelusi, A. Passarella, and M. Conti. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *IEEE Communications Magazine*, 44(11), November 2006.

[90] T. Pering, Y. Agarwal, R. Gupta, and R. Want. CoolSpots: reducing the power consumption of wireless mobile devices with multiple radio interfaces. In *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2006.

[91] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[92] D. Petrovic, R. C. Shah, K. Ramchandran, and J. Rabaey. Data Funneling: Routing with aggregation and compression for wireless sensor networks. In *Proceedings of the International Workshop on Sensor Network Protocols and Applications (SNPA)*, 2003.

[93] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[94] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica. A unifying link abstraction for wireless sensor networks. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[95] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN), Special Track on Platform Tools and Design Methods for Network Embedded Sensors (SPOTS)*, 2005.

[96] S. S. Pradhan, J. Kusuma, and K. Ramchandran. Distributed compression in a dense microsensor network. *IEEE Signal Processing*, March 2002.

[97] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[98] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin. Sympathy for the sensor network debugger. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[99] N. Ramanathan, M. Yarvis, J. Chhabra, N. Kushalnagar, L. Krishnamurthy, and D. Estrin. A stream-oriented power management protocol for low duty cycle sensor network applications. In *Proceedings of the Workshop on Embedded Networked Sensors (EmNets)*, 2005.

[100] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *SIAM Journal of Applied Mathematics*, 8:300–304, 1960.

[101] N. Sadagopan, F. Bai, B. Krishnamachari, and A. Helmy. PATHS: analysis of path duration statistics and their impact on reactive manet routing protocols. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003.

[102] C. M. Sadler and M. Martonosi. Data compression algorithms for energy-constrained devices in delay tolerant networks. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.

[103] C. M. Sadler and M. Martonosi. DALi: A communication-centric data abstraction layer for energy-constrained devices in mobile sensor networks. In *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2007.

[104] P. Samar and S. B. Wicker. On the behavior of communication links of a node in a multi-hop mobile environment. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2004.

[105] T. Schmid, T. Dreier, and M. B. Srivastava. Software radio implementation of short-range wireless standards for sensor networking. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.

[106] Nokia sensorplanet. http://www.sensorplanet.org.

[107] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: Modeling a three-tier architecture for sparse sensor networks. In *Proceedings of the International Workshop on Sensor Network Protocols and Applications (with IEEE ICC '03)*, 2003.

[108] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: modeling and analysis of a three-tier architecture for sparse sensor networks. *Ad Hoc Networks*, 1(2-3):215–233, 2003.

[109] V. Shnayder, M. Hempstead, B. rong Chen, G. W. Allen, and M. Welsh. Simulating the power consumption of large-scale sensor network applications. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[110] T. Small and Z. 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 ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003.

[111] D. Son, B. Krishnamachari, and J. Heidemann. Experimental study of concurrent transmission in wireless sensor networks. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.

[112] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis. Understanding the causes of packet delivery success and failure in dense wireless sensor networks. Technical Report SING-06-00, Stanford University, 2006.

[113] K. Srinivasan and P. Levis. RSSI is under appreciated. In *Proceedings of the Workshop on Embedded Networked Sensors (EmNets)*, 2006.

[114] F. Stann and J. Heidemann. RMST: Reliable data transport in sensor networks. In *Proceedings of the International Workshop on Sensor Network Protocols and Applications (SNPA)*, 2003.

[115] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[116] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin. Habitat monitoring with sensor networks. *Communications of the ACM*, 47(6):34–40, 2004.

[117] Tinyos. https://www.tinyos.org/.

[118] G. Tolle and D. Culler. Design of an application-cooperative management system for wireless sensor networks. In *Proceedings of the European Workshop on Wireless Sensor Networks (EWSN)*, 2005.

[119] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A macroscope in the redwoods. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[120] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh. Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks. In *Proceedings of the IEEE INFOCOM*, 2002.

[121] Uucp mail interchange format standard. http://www.ietf.org/rfc/rfc976.txt.

[122] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University, 2000.

[123] VFML Toolkit, http://www.cs.washington.edu/dm/vfml/main.html.

[124] L. Viennot, P. Jacquet, and T. H. Clausen. Analyzing control traffic overhead versus mobility and data traffic activity in mobile ad-hoc network protocols. *Wireless Networks*, 10(4):447–455, 2004.

[125] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy. Pump-Slowly, Fetch-Quickly (PSFQ): A reliable transport protocol for sensor networks. *IEEE Journal on Selected Areas in Communications*, 2005.

[126] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell. CODA: congestion detection and avoidance in sensor networks. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[127] C.-Y. Wan, S. B. Eisenman, A. T. Campbell, and J. Crowcroft. Siphon: Overload traffic management using multi-radio virtual sinks. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[128] Y. Wang, M. Martonosi, and L.-S. Peh. MARio: Mobility-adaptive routing using route lifetime abstractions in mobile ad hoc networks. *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R)*, 8(4), 2004.

[129] Y. Wang, P. Zhang, T. Liu, C. Sadler, and M. Martonosi. CRAWDAD data set princeton/zebranet (v. 2007-02-14). Downloaded from http://crawdad.cs.dartmouth.edu/princeton/zebranet, Feb. 2007.

[130] H. Weatherspoon and J. Kubiatowicz. Erasure coding vs. replication: A quantitative comparison. In *Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[131] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2006.

[132] G. Werner-Allen, P. Swieskowski, and M. Welsh. MoteLab: A wireless sensor network testbed. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN), Special Track on Platform Tools and Design Methods for Network Embedded Sensors (SPOTS)*, 2005.

[133] J. Widmer and J.-Y. L. Boudec. Network coding for efficient communication in extreme networks. In *Proceeding of the ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN)*, 2005.

[134] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, 2nd Edition*. Morgan Kaufmann, 2005.

[135] Wizzy digital courier. http://www.wizzy.org.za/.

[136] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[137] Imote2 data sheet. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Imote2_Datasheet.pdf.

[138] N. Xu, K. K. Chintalapudi, and D. Ganesan. A wireless sensor network for structural monitoring. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[139] Y. Xu and W.-C. Lee. Exploring spatial correlation for link quality estimation in wireless sensor networks. In *Proceedings of the Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM)*, 2006.

[140] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the IEEE INFOCOM*, 2002.

[141] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *Proceedings of the IEEE INFOCOM*, 2003.

[142] J. Yoon, B. D. Noble, M. Liu, and M. Kim. Building realistic mobility models from coarsegrained traces. In *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2006.

[143] H. Zhang, A. Arora, and P. Sinha. Learn on the fly: Data-driven link estimation and routing in sensor network backbones. In *Proceedings of the IEEE INFOCOM*, 2006.

[144] P. Zhang, C. M. Sadler, S. Lyon, and M. Martonosi. Hardware design experiences in Zebranet. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[145] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[146] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2004.

[147] R. Zheng, J. C. Hou, and L. Sha. Asynchronous wakeup for ad hoc networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003.

[148] N. Zhou, H. Wu, and A. A. Abouzeid. Reactive routing overhead in networks with unreliable nodes. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2003.