4-23-2013

Collaborative Communication And Storage In Energy-Synchronized Sensor Networks

Mingsen Xu

COLLABORATIVE COMMUNICATION AND STORAGE IN ENERGY-SYNCHRONIZED SENSOR NETWORKS

by

Mingsen Xu

Under the Direction of WenZhan Song

ABSTRACT

In a battery-less sensor network, all the operation of sensor nodes are strictly constrained by and synchronized with the fluctuations of harvested energy, causing nodes to be disruptive from network and hence unstable network connectivity. Such wireless sensor network is named as energy-synchronized sensor networks. The unpredictable network disruptions and challenging communication environments make the traditional communication protocols inefficient and require a new paradigm-shift in design. In this thesis, I propose a

set of algorithms on collaborative data communication and storage for energy-synchronized sensor networks. The solutions are based on erasure codes and probabilistic network codings. The proposed set of algorithms significantly improve the data communication throughput and persistency, and they are inherently amenable to probabilistic nature of transmission in wireless networks.

The technical contributions explore collaborative communication with both no coding and network coding methods. First, I propose a collaborative data delivery protocol to exploit the optimal performance of multiple energy-synchronized paths without network coding, i.e. a new max-flow min-variance algorithm. In consort with this data delivery protocol, a localized TDMA MAC protocol is designed to synchronize nodes' duty-cycles and mitigate media access contentions. However, the energy supply can change dynamically over time, making determined duty cycles synchronization difficult in practice. A probabilistic approach is investigated. Therefore, I present Opportunistic Network Erasure Coding protocol (ONEC), to collaboratively collect data. ONEC derives the probability distribution of coding degree in each node and enable opportunistic in-network recoding, and guarantee the recovery of original sensor data can be achieved with high probability upon receiving any sufficient amount of encoded packets. Next, OnCode, an opportunistic in-network data coding and delivery protocol is proposed to further improve data communication under the constraints of energy synchronization. It is resilient to packet loss and network disruptions, and does not require explicit end-to-end feedback message. Moreover, I present a network Erasure Coding with randomized Power Control (ECPC) mechanism for collaborative data storage in disruptive sensor networks. ECPC only requires each node to perform a single broadcast at each of its several randomly selected power levels. Thus it incurs very low communication

overhead. Finally, I propose an integrated algorithm and middleware (Ravine Stream) to improve data delivery throughput as well as data persistency in energy-synchronized sensor network.

INDEX WORDS: Collaborative network coding, Probability distribution, Opportunistic routing, Disruptive sensor networks

COLLABORATIVE COMMUNICATION AND STORAGE IN ENERGY-SYNCHRONIZED SENSOR NETWORKS

by

MINGSEN XU

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2013

COLLABORATIVE COMMUNICATION AND STORAGE IN ENERGY-SYNCHRONIZED SENSOR NETWORKS

by

MINGSEN XU

Committee Chair: WenZhan Song

Committee: Xiaolin Hu

Yingshu Li

Yichuan Zhao

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

May 2013

DEDICATION

This dissertation is dedicated to Georgia State University.

ACKNOWLEDGEMENTS

My doctorate venture would not be complete without the support of many people. First, I want to express my sincere gratitude to my advisor Professor WenZhan Song, for his generous financial support throughout my PhD study. This dissertation would not be possible without his insightful guidance and patience over the last four and half years. He shares his great research experience and passion with me. He not only motivates me to learn and grow along the phd course, but also surpass him in the future. It is not only his invaluable academic knowledge and methodologies, but also his passionate attitude and discipline to succeed my future career development.

I am also very grateful to my parents for their unconditional love and support for all these years, even I am barely around. My entire life would not be considered perfect until I met my wife, ChuiYing (Eunice) Law in 2009. She shows me great resolution and determination, and inspires me every single day. Every time I struggle and look for the support, she is always there to be with me, supporting and encouraging me throughout the tough moments. My euphoric time is filled with much more joyful memory with her presence. I thank her and love her from the bottom of my heart.

I also made many great friends at both Washington State University and Georgia State University during my PhD endeavor. Renjie Huang, Gang Lu and Xiaogang Yang, we are involved in the same research team at WSU, live in the same apartment, play and hang out all together for most of time. The joyful time reminds me of the beautiful nature scene and atmosphere of Vancouver Washington. I am happy for the graduation and good career development of them. They are still available for chatting from time to time. Debraj De, Lei Shi, Song Tan, Qinjun Xiao and Lei Zhang, we start our research journey in a brand new environment - GSU in 2010. During past two years, all of them show me passion, perseverance and ability on finding and solving difficult problems. I would like to thank them for their superior intelligence and time spent on helping me with countless discussions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

- LT - Luby Transform
- ISD - Ideal Soliton Distribution
- RSD - Robust Soliton Distribution
- WSN - Wireless Sensor Network
- TDMA - Time Division Multiple Access
- MAC - Media Access Control
- CTP - Collection Tree Protocol
- LDPC - Low Density Parity Check
- EXOR - Extreme Opportunistic Routing
- ONEC - Opportunistic Network Erasure Coding
- OnCode - Opportunistic in-Network Coding
- ECPC - Erasure Coding with randomized Power Control
- PRR - Packet Reception Rate
- HMM - Hidden Markov Model
- XOR - Exclusive OR
- TCP - Transport Control Protocol
- TOSSIM - TinyOS Simulator
- QoS - Quality of Service

- ASE - Adaptive Source Encoding
- OINE - Opportunistic In-Network Encoding
- EFU - Encode and Forward Utility
- PMS - Power Management System
- ESM - Energy Synchronization Module
- EWMA - Exponentially Weighted Moving Average
- EDFC - Exact Decentralized Fountain Codes
- RCDS - Raptor Codes based Distributed Storage

CHAPTER 1

INTRODUCTION

1.1 Introduction

Sensor networks consist of spatially distributed devices communicating with radios and cooperatively sensing physical or environmental conditions. They have emerged as a key technology to enable many critical social and military applications. It is especially useful in catastrophic or emergency scenarios, such as floods, fires, volcanos, battlefields, where human participation is too dangerous and infrastructure networks are impossible or too expensive.

1.1.1 Challenges from Environment

Sensor networks provides people with great facility to enable different applications in various challenging environments. However, those challenging environments also pose significant challenges to network sustainability and reliability at the same time. For example, in a volcano monitoring application [1], the occasional eruptions, rock avalanches, landslides, earthquakes, gas/steam emissions, as well as harsh weather conditions, often exacerbate link qualities and even destroy stations. In a battleeld surveillance application [33], the sensors may be jammed, hijacked or destroyed by the adversaries. In the mountainous forest environment, the communication link bandwidth and reliability is severely limited by leaves, branches, and rugged terrain. In addition, weather conditions, such as heavy rain, wind, and snow, may further challenge network connectivity. In those challenged sensor networks, a predictable and stable path may never exist, the network connectivity is intermittent, and a node could suddenly appear or disappear. The link quality may vary significantly and asymmetric links are common. The rare upload opportunity and unpredictable node disruptions often result in data loss. Since the unpredictable node disruptions often result in data packet lost, which makes traditional reliable data transfer protocols infeasible. On particular, the

lost of important ACK (acknowledgment) or NACK (negative acknowledgment) message can cause message implosion disaster in reliable data transfer. This demands a new design paradigm to collaboratively utilize the limited communication, energy and storage resources to combat network disruptions and maintain reliable operations.

1.1.2 Challenges from Energy Harvesting

Renewable energy holds great promise of making wireless sensor network truly battery-less. It will enable sensor nodes to sense, compute and communicate for a prolonged and even perpetual lifetime. However, the amount of harvested energy has dynamic and volatile patterns, since it depends on many uncertain environmental factors (such as weather, light intensity, temperature etc.). *Energy synchronization* is the key to meet the lifetime requirement in battery-less sensor networks, preventing the residual energy from being overdrawn. The term "*energy synchronization*" means that total energy consumption due to node operations (including radio communication, sensing and computing) need to match the dynamic harvested energy or residual energy constraints.

In the literature, several works (such as in [2]) adapt radio duty cycle in MAC protocol to different energy constraints. Other existing methodologies, such as opportunistic routing and joint energy-routing optimization were proposed to deliver data collaboratively. Opportunistic routing is adopted [3–5] to take advantage of multiple transmission opportunities to improve data throughput. Opportunistic routing allows packet overhearing. Any node closer to sink and receiving packet, has opportunity to forward the packet. Random linear coding is combined to make full use of spatial diversity in the work of [4, 5]. The reduced communication overhead further enhances the network throughput. Source nodes demand explicit ACK feedback message from decoder to advance encoding blocks. The reliance on ACK message for reliable data delivery makes it difficult in such a disruptive multi-hop network. More importantly, existing solutions are unaware of dynamic energy fluctuations in nodes. The obliviousness of energy may cause nodes to fail in practice. Joint optimization [6] considers energy and routing together for the decisions and gives an optimal solution. But

joint optimization requires determined communication pattern and energy information and intensive computation cost, which prevents it from being conducted dynamically on sensor nodes. Designing efficient data delivery protocol, which fully take advantages of both heterogeneous nodal energy constraints and transmission opportunities, remains under-explored. An appropriate protocol design needs to allow for the unpredictable disruptions due to nodal energy constraints, and at the same time utilize opportunistic transmission with minimum cost. Such a challenging data communication environment makes the traditional data delivery protocol insufficient, thus it requires a paradigm-shifting design.

1.2 Our Approach

The technical contributions explore collaborative communication with both no coding and network coding methods. First, I propose a collaborative data delivery protocol to exploit the optimal performance of multiple energy-synchronized paths without network coding, i.e. a new max-flow min-variance algorithm. In consort with this data delivery protocol, a localized TDMA MAC protocol is designed to synchronize nodes' duty-cycles and mitigate media access contentions. However, the energy supply can change dynamically over time, making determined duty cycles synchronization difficult in practice. A probabilistic approach is investigated. Therefore, I present Opportunistic Network Erasure Coding protocol (ONEC), to collaboratively collect data. ONEC derives the probability distribution of coding degree in each node and enable opportunistic in-network recoding, and guarantee the recovery of original sensor data can be achieved with high probability upon receiving any sufficient amount of encoded packets. Next, OnCode, an opportunistic in-network data coding and delivery protocol is proposed to further improve data communication under the constraints of energy synchronization. It is resilient to packet loss and network disruptions, and does not require explicit end-to-end feedback message. Moreover, I present a network Erasure Coding with randomized Power Control (ECPC) mechanism for collaborative data storage in disruptive sensor networks. ECPC only requires each node to perform a single broadcast at each of its several randomly selected power levels. Thus it incurs very

low communication overhead. Finally, I propose an integrated algorithm and middleware (Ravine Stream), illustrated in Figure 1.1 to improve data delivery throughput as well as data persistency in energy-synchronized sensor network.

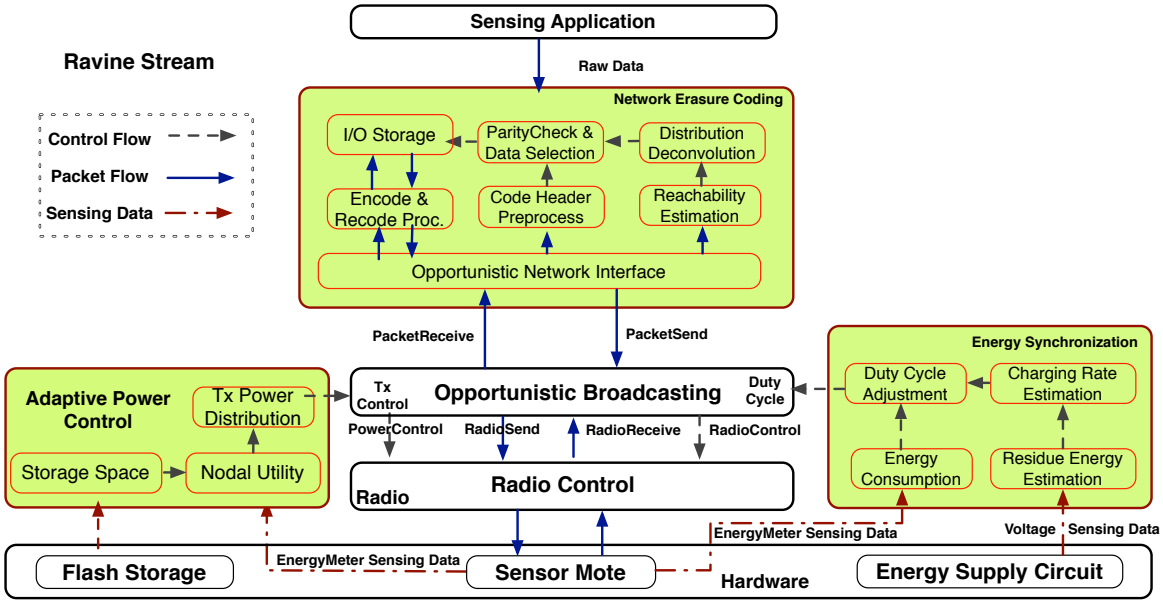


Figure 1.1 Protocol Implementation of Ravine Stream

The rest of thesis is organized as follows. Chapter 2 describes the related works in the literature. Chapter 3 discuss the collaborative data delivery without network coding, exploring the optimal data delivery paths and synchronizing nodes' duty cycles. Chapter 4 introduces opportunistic network coding method to collaboratively collect data and improve data communication throughput. Chapter 5 propose OnCode by considering the real energy constraints in sensor motes. And chapter 6 describe ECPC distributed data storage to preserve data persistency. Chapter 7 integrates the collaborative communication and storage into the final solution, "Ravine Stream", for energy-synchronized sensor networks. Chapter 8 concludes this work.

CHAPTER 2

RELATED WORKS

2.1 Optimal Data Rate Assignment in Data Communication

The max-min optimal rate assignment for network fairness have been studied in a few related works. In [7], it presents an iterative linear programming solution, which has the same goal as our max-flow and min-variance algorithm. They proved that there is one and only one such assignment for a given configuration of the sensor networks. Its worst-case time complexity is $O(|V|^2C(|E|))$, where $C(|E|)$ is the complexity of linear programming with $O(|E|)$ variables and constraints. The complexity of $C(|E|)$ is not given in the paper, however, it is known to be polynomial $O(|E|^\alpha)(\alpha \geq 1)$, which is at least $O(|E|)$. Recall that the time complexity of our max-flow and min-variance algorithm is bounded by $O(|V|^2|E|)$ which is lower. [7] did not provide any simulation or test bed results. In [8], it aims at designing a solution for fair and high throughput data extraction from all nodes in the presence of renewable energy sources. It gives both a centralized algorithm and distributed algorithm. The centralized algorithm is similar to [7], except that they assume resources are dynamic and vary over the time. However, the asynchronous distributed algorithm assumes that the data collection flow is a tree structure, so it may not achieve maximum throughput as discussed earlier in the paper.

A follow-up work [6] from the same group further considers resource dynamic over time in a rechargeable sensor network and proposes to adapt the sampling rate and route with the objective of maintaining the battery at a target level. Some related works have studied congestion control and fair rate assignment schemes. [9] proposes a distributed rate allocation scheme (IFRC) without end-to-end reliability. IFRC scheme detects the congestion point, and advertises the congestion by overhearing each other's messages, finally converges to a rate assignment, which is fair and avoids congestion collapse. A more recent work [10] presents

a reliable centralized algorithm (RCRT) to allocate the fair date rate to each node, which can achieve more than twice the rate achieved by IFRC. Those works are not optimized for heterogeneous duty-cycled networks, where optimization in MAC and routing layer is necessary. Our work differs from the aforementioned work: they are designed as transport protocols for sensor network which are evaluated on either static routing trees or dynamic routing trees. However, our proposed algorithm, involving network and MAC protocols, can adjust fair flows on multiple routing paths, and achieve the maximum flow and fairness simultaneously.

2.2 Duty-cycled MAC Protocol

The related works on MAC protocol design for duty-cycled sensor networks are plentiful. A common goal is to reduce the energy consumption, while increasing the possibilities of rendezvous between sender and receiver. [11] provides a distributed transmission slot scheduling protocol to save power consumption in wireless sensor networks. But the control message would be appreciable under large sensor network, and "hidden terminal" could cause the collision of advertisement or confirmation messages. In B-MAC [12] and X-MAC [13], preambles are used by sender to wake up receiver. In contrast, RI-MAC [14] let the receiver wake up the sender to transmit and can avoid buffer overflow and data losses. The current consumption of preambles or control messages is not necessarily small. Our proposed algorithm differs in synchronizing the wake-up scheduling by allocating wake-up time slot based on MaxFair flow ID and nodes' hop-count, reducing the energy consumption on wake-up scheduling.

[2] designs a localized energy synchronization control mechanism that shuffles or adjusts the working schedule of a node to optimize delays in the presence of varying energy budget over time. In their work, a delay model for cross-traffic at individual nodes is derived and a stair effect is observed in low-duty-cycle networks. More recently, [15] proposes a proactive solutions for minimizing sink-to-node communication delay in energy-harvesting sensor networks. They introduce a delay maintenance algorithm with minimum energy

consumption. In both works, they assume node duty-cycles can be augmented to reduce delays. We note that in an energy-synchronized network, each node's duty-cycle shall be determined by its residual energy (or energy harvesting rate) and augmenting duty-cycles may sacrifice lifetime. Therefore, we utilizes multiple paths to minimize delays.

2.3 Erasure Codes

Erasure codes enable the recovery of k symbol from a subset of n symbols. It has been employed in many applications. For example, the most widely applied erasure codes is Reed-Solomon codes. Additionally, LDPC [16] codes and fountain codes were also designed to allow for low complexity and computation cost encoding and decoding.

LT Codes [17] is the first effective implementation of erasure codes. LT Codes propose a Robust Soliton Distribution (RSD), which is essential to the success of decoding process. The theoretical analysis guarantees that the decoding can succeed with high probability under the proposed degree distribution. Raptor Codes [18] is designed to yield a encoding solution of less packet complexity. Raptor codes reduce the degree to a constant factor of raw symbols size by inserting a pre-code stage before LT codes. Though Raptor codes requires a less decoding complexity, the pre-code stage is not rateless and lost packets are possible to fail the decoding. LT codes are completely rateless coding schemes, in which random encoded packets are independent from each other and lost packets do not compromise the entire decoding. In other words, it can be applied to combat the disruptive network communications. Therefore we focus on the LT codes and design a LT codes based network coding to cope with data collection in disruptive sensor networks.

2.4 Micro Solar Power System

Recent research works on micro-solar system have shown the potential of driving low-power devices without battery, such as Prometheus [19], AmbiMax [20], Heliomote [21] solar system, and Trio [22]. The work in [23] proposes a model and guideline for analyzing the design of micro-solar system, based on the empirical study of Heliomote and Trio system.

The work in [24] proposes a leakage-aware energy control layer to adapt the operation of application to harvesting energy. [25] proposes energy-harvesting low-power device, EnHANTs. [26] study how to allocate energy spending rate with various predictable energy inputs. Our contribution is proposal of a coding-based network protocol that can adaptively utilize the energy from micro-solar power system. It is evaluated on a real micro-solar powered indoor testbed.

Based on the solar-powered devices, SolarStore in [27] provides a storage-centric service, which adaptively balances between the data reliability and data sensing. SolarCodes [28] maximize the usage of surplus energy by adjusting the redundant factor of erasure coding in each link respectively. OnCode protocol is distinguishable from two perspectives: (1) OnCode synchronizes data delivery with any energy constraints, not only of surplus energy, but also of insufficient energy. (2) OnCode exploits opportunistic routing instead of a predetermined routing, which reduces the coding overhead significantly. The diversity and randomness in opportunistic coding improve quality of data delivery by enhancing flow throughput and fairness.

2.5 Network Coding in Collaborative Data Communication

DCAR [29] discover available paths and detect potential network coding opportunities between a given source and destination, eliminating two-hop coding limitation. It also proposes a routing metric CRM to compare "coding-possible with "coding-impossible paths. However, DCAR is based the "best-path" routing, without considering lossy link and dynamic characteristics of wireless channel. In disruptive wireless networks, the coding opportunity on a path could vary according to different traffic and link quality. CORE in [30] proposes a coding-aware opportunistic routing protocol. It improves the link-level data delivery reliability and network throughput by combing hop-by-hop opportunistic forwarding and localized inter-flow network coding.

The aforementioned schemes are deterministic network coding, where coding opportunities are identified beforehand, and nodes are assigned with fixed coding responsibilities.

Thereafter, they are heavily reliant on the topology of the network and it may be a challenge in disruptive wireless network.

Distributed LT Codes [31, 32] gives a data relay model, based on which it derives individual degree distribution for each source node one hop away from relay. The degree deconvolution can yield a received packet degree distribution closely approaching to RSD. We distinguish our work by designing a network coding that not only apply to a more general data collection structure, but also employ the opportunistic routes to construct the random structure in disruptive network.

There are another recent work [33] which applies LT codes to network coding for data collection in wireless sensor networks. In LTNC [33], if a node needs to generate a recoded packet, it first generates an encoded packet of degree d, where d is drawn from a Robust Soliton Distribution, selecting the packets in the buffer; second, it refines the obtained packet so that the variance of the distribution of degrees for native symbol is reduced. LTNC maintains the RSD property of encoded packet in a decentralized way which in turns make a decoding process of low computation cost. However, it introduces the intermediate coding latency and requires a considerable amount of memory space to store the received packets for future encoding. Moreover, it does not design and evaluate the scheme in the context of disruptive sensor network. To the best of our knowledge, our proposed ONEC protocol is the first coding scheme to solve the problem of achieving reliable data transmission by applying low-complexity network coding in disruptive communication network.

The opportunistic routing proposed in ExOR [3] utilizes the probabilistic receiving in multiple hops distance to accelerate the packet forwarding. MORE [4] uses random linear coding to mix packets before forwarding them. It has advantages over the ExOR [3], which is first routing protocol with opportunistic forwarding. MORE [4] takes advantages of spatial reuse by random linear coding to ease the problem of forwarder selection in ExOR, and improve the data delivery quality. CCACK [5] utilizes a Null-Space Based (NSB) message to acknowledge the reception of encoded packets, which suppress not only redundant packet retransmission, but also the non-innovative encoded packets. It is because that neighbors,

hearing NSB messages, can determine if it can generate another linearly independent packets for its neighbors. Though CCACK improves MORE by suppressing overhead packet from transmission, the time of moving to next data segment still relies on the ACK from destination. Observing the delay introduced in data delivery of segmented network coding, SlideOR in [34] explores sliding window mechanism. The random linear coding is employed in SlideOR, since the decoder can determines the up-to-date decodable data by Gaussian Elimination of received packets. However, SlideOR is sensitive to ACK notification from decoder to advance encoding window, which would be impractical in disruptive communication environments. SlideOR does not consider spatial diversity and opportunistic coding to improve coding gain and quality of data delivery. GROWTH [35] achieves the data reliability in a catastrophic network. The main contribution is that growth codes increase its code degree over time, which maximize the partial decoding probability at any given time during the decoding process. We evaluate and compare our ONEC schemes with these two network coding in section 7.2.

The difference of our OnCode work from previous approaches are: *first*, we use erasure codes, which fully benefits from the low complexity of belief propagation decoding. *Second*, OnCode maximizes the space diversity and probabilistic packet transceiving by opportunistic in-network encoding. ONEC [36] is the work close to OnCode, which explores network erasure coding in disruptive network. However, ONEC requires a collection tree structure to conduct recursive deconvolution of degree distribution, which costs considerable overhead whenever updating. OnCode conduct deconvolution based on distributed packet reachability estimation, with low communication overhead. Moreover, ONEC costs more overhead and can not adapt to network dynamics well, especially in energy-synchronized sensor network where the energy resource are dynamic volatile over time. OnCode self-tunes the data coding and probabilistic forwarding *in-situ* with energy distribution and variations.

2.6 Network Coding in Collaborative Data Storage

In [37], Dimakis et al. propose a class of distributed erasure codes to solve the data collection query problem. The contribution of paper is to prove that $O(ln(k))$ "pre-routed" packets for each data node is sufficient such that collecting packets from any k of n storage nodes can retrieve raw data with high probability. Essentially, Dimakis et al.'s contribution is to make this random bipartite graph as sparse as possible by constructing a perfect matching w.h.p., which ensure the codes are decodable by proof. Another interesting point in [37] is the connection between distributed erasure codes and network codings. Distributed erasure codes can be viewed as the linear network codes between data nodes and storage nodes placed in distinct parts of a random bipartite graph. There is no explicit routes between data nodes and storage nodes, because the packets are randomly generated. In [38], Dimakis et al. studied the problem of establishing fountain codes for distributed storage and data collection. The technical contribution of [38] is a degree distribution for fountain codes, which enables the "pre-routing" degree of each data node is bounded by a constant a.a. (almost always). Later a randomized algorithm is also proposed to find the random "pre-route" between each encoded packet and raw data on a given grid topology.

[39] provides a LT codes based network coding to solve the data persistence problem in a large-scale network. The contribution is to find random routes between data node and encoded packet using random walks. Each data node disseminate a constant number of packet to the network which will stop in specific storage node with probability computed by its selected RSD (Robust Soliton Distribution) degree. To ensure the stop probability closely approach to the desired degree distribution in the network, [39] utilized the Metropolis algorithm to construct a transition probability for forwarding packets on random walks. Nevertheless, max node degree is required for constructing Metropolis transition matrix, which is hard to obtain in large-scale network, especially when topology changes. In [40], Aly et al. designed two distributed data dissemination algorithms based on LT codes, which eliminate the global knowledge of maximum node degree. In the first algorithm, network size

n and number of raw symbol k is still assumed available in each local node, while the n and k is obtained by estimating the random walk data dissemination in the second algorithm. Later in [41], Aly et al. extend to develop Raptor codes based coding scheme, which reduce the packet demand from logarithmic to constant ϵ by inserting a proper pre-code in front of LT codes. Most recently, a survey [42] provides a summary of research problems and results in maintaining the reliability of distributed storage system by partially repairing the encoded packets in poor nodes. This repairing demand a partial recovery of the replaced codes, while the distributed network coding mentioned above focus on the full decoding of the raw symbols from a sufficient subset of codes. In [38], Dimakis et. al. presented a new degree distribution of fountain codes for distributed storage and data collection. It enables the "pre-routing" degree of each data node to be bounded by a constant number almost always. Later an algorithm is also proposed to find the random "pre-route" between each encoded packet and raw data on a given grid topology. [43] proposes a "packet-centric" approach for distributed data storage. It is also based on random walk mechanism. Every rateless packet is initiated with a selected code degree. While the rateless packet randomly traverses through the network, it collects the exact same number of encoding symbol from uniformly distributed sensor data as its code degree, and terminates in a random node, which may require different storage spaces at different nodes.

However, in those random walk based schemes [38–41, 43], the message cost is significant. Our works reduce message cost during the codewords construction period to $O(1)$ per node. Each node only needs to perform several rounds of broadcast, and each node just needs to encode several randomly heard packets to a single encoded packet at each period. It also implies that the storage space requirement is uniform across the network. It has a fast termination time, which means more resilience to network disruptions as well.

Distributed data storage proposed in [44, 45] maximize network storage capacity by offloading local data to network when memory overflow. However, they do not consider disruptive network conditions. In [46], the proposed scheme replicates data items on mobile hosts to improve the data accessibility in case of partitioned network, by considering mobile

users' access behavior, and read/write patterns. SolarStore in [27] maximizes the retrieval data under energy and storage constraints. The proposed approach can dynamically adjust the degree of data replication according to the energy and storage.

Recently, applying network coding to preserve the data persistence against disruptive node conditions is a subject of increasing research interest [47–50]. The advantage of network coding is to increase the diversity and redundancy of data packets so as to improve data persistence. [35] proposes Growth Codes to preserve data persistence in a disruptive sensor network. Growth Codes increase their code degrees on-the-fly efficiently as data is collected in the gateway. The changing point of code degree is designed to maximize data persistence when decoding. [51] proposes geometric random linear codes, to encode data in a hierarchical fashion in geographic regions with different sizes. It enables to locally recover small amount of node failures. In [52], Albano et al. construct a random linear coding for distributed data storage with near linear message cost. In particular, the paper employs spatial gossip to construct the linear codes. The idea is that each node i chooses another node j with probability proportional to $1/|ij|^3$, where $|ij|$ is the Euclidean distance between i and j. It is proved that a data symbol from node i is delivered to a node j with the probability $1-O(1/n)$ after $O(log^{3.4}n)$ iterations. Thus, using spatial gossip constrains the total number of transmissions during codes construction to $O(n\ polylogn)$. The above methods explore random linear coding, which requires data packets to traverse through network to generate the linear codes. Our ECPC approach only needs to encode data received from localized broadcast.

Tang et al. [53] formalizes storage depletion induced data redistribution as the minimum cost flow problem, and devises a distributed data redistribution algorithm (PoF). [54] consider the energy and storage depletion to maximize data preservation time. Valero et al. [55] formulates the problem as an optimization problem and use Linear Programming to find the optimal solution. A distributed algorithm ($EDR2$) is implemented for in-network storage and later data retrieval. [56] proposes a probabilistic broadcasting based data redistribution scheme. Their contribution is to derive a proper probability distribution for rebroadcasting

packet. Receivers encode data using LT codes and store them, so that a collector can retrieve data by visiting small set of data. Hou et al. [57] seeks to maximize the minimum remaining energy among the nodes storing data items. It presents a centralized greedy heuristic to approximate the optimal solution under certain conditions.

The above optimization algorithms, though considering energy and storage constraint, ignore an important fact that disruptive condition may happen in the course of data redistribution. The disruptive network seriously challenges data redistribution. Distinct from the existing works, our contributions of *Ravine Stream* are three fold: *first*, *Ravine Stream* conducts probabilistic broadcasting with adaptive transmission power to overcome the disruptive connection during data redistribution. It has high energy efficiency and low message redundancy. *Second*, *in-situ* recoding can reduce data redundancy in symbol wise. *Third*, *Ravine Stream* generalizes the energy and storage constraints to nodal utility, which includes failure probability and storage constraint. According to the nodal utility, data storage decision can be made distributively.

CHAPTER 3

COLLABORATIVE DATA COMMUNICATION WITHOUT CODING

3.1 Problem Statement and Motivation

In a duty-cycled sensor network, sensor node can stay in any of the three states: *transmitting*, *receiving* and *sleeping*. Given a node u, we denote T_u, R_u and S_u as its total time duration of transmitting, receiving and sleeping respectively in the entire lifetime. Clearly, the transmitting and receiving data rate of node u is proportional to the active time duration of transmitting and receiving. If link loss ratio is also considered, then T_u and R_u represent the effective transmission and receive data rate of node u. Then the node u's capacity $C_u = (T_u + R_u) \cdot r$ denotes the total flow node u can handle, where r is the radio data rate in each node. Let D_u be the bandwidth demand of node u's own data, then the flow conservation $T_u = R_u + \frac{D_u}{r}$ shall hold to avoid data losses. To meet lifetime requirement, C_u shall be synchronized with the usable energy of each node u. Under the heterogeneous constraints on node capacity C_u, network nodes need to find T_u and R_u for each node u, such that the network throughput is maximized and flow fairness is guaranteed. We solve this through the following max-flow and min-variance algorithm.

In a heterogeneous duty-cycled sensor network, a data collection tree for data delivery is not necessarily the best solution. Each node may use multiple paths to deliver data for itself and its upstream nodes. Clearly, multiple paths may provide nodes with more delivery opportunities. In Figure 3.1, we show a sensor network with heterogeneous node capacities. It can be clearly observed that using collaborative "multi-path" can deliver more data than using only one single path. To be more specific, in Figure 3.2, we illustrate that there are 4 starved nodes if a collection tree is used. Here, we assume each node has one unit flow, which applies to the rest figures and examples in this paper. A node is starved when its own unit flow is held due to the bottleneck in the downstream nodes. In Figure 3.2, node

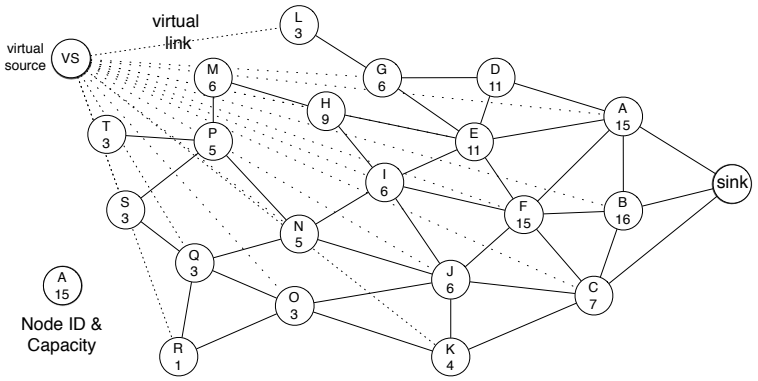


Figure 3.1 Illustration of original network topology and node capacity. In a many-to-one data collection network, a Virtual Source (VS) can be added to apply max network flow algorithm.

T and S are starved, because their unit flows are throttled due to the bottleneck in node I. Compared with the Figure 3.3, it is easy to see that collaborative "multi-path" can deliver more data than a single path. And it motivates us to fully utilize the collaborative data delivery paradigm to improve data throughput. Then, the challenge is how to find the optimal paths for each individual node and how to manipulate the data flows such that network throughput and flow fairness are maximized, assuming network nodes' duty-cycles are fixed but heterogeneous. Next, the problem of finding max delivery throughput can be converted to the max flow problem. To solve the max flow problem in this converged tree topology, a virtual source is added and connected to each source node, which transforms the "many-to-one" flow maximization problem to "one-to-one" flow maximization problem.

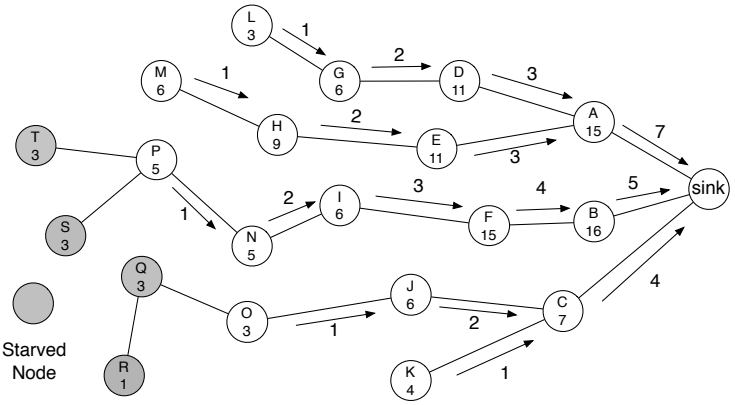


Figure 3.2 Illustration of data collection tree, where 4 nodes are starved.

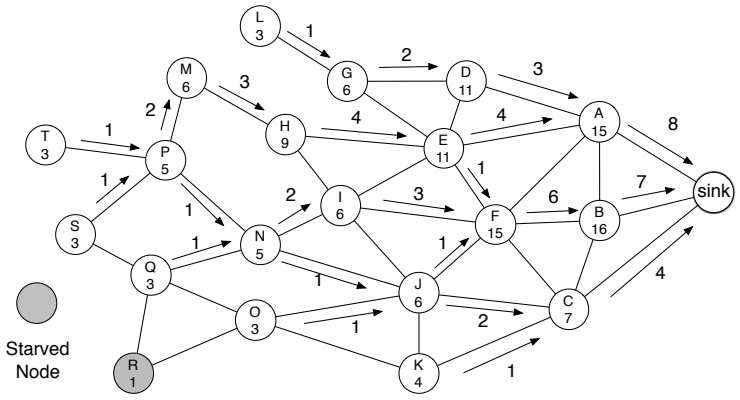


Figure 3.3 Illustration of max flow network generated by our "loop-free" max-flow algorithm, where only 1 node is starved. In this and following examples, we assume each node has unit flow demand.

We formally state our throughput maximization problem. Given a sensor network $G = \{V, E\}$, where V is the set of nodes and E is the set of links. Each node is subject to its energy capacity constraint: $C : V \rightarrow \mathbb{R}^+$. Further, let $f_u(v, w)$ denote the flow from node v to w originated at source node u. Our objective is to find the optimal flow scheduling: $\{f_u(v, w)\}$, such that the network throughput ($F = \sum_{x \in V \setminus sink} f_x(x, sink)$) is maximized, and the variance of originating flow for each node ($f_u(*, sink)$) is minimized.

Table 5.1 shows the notations used in the paper.

Table 3.1 List of notations in max-flow and min-variance algorithm

C_u	capacity of node u
D_u	source flow demand of node u
N_u	neighbors of node u
r	radio data rate
$h[u]$	height function of node u
E_f	edge set in the residual graph
$\delta_f(u, v)$	distance between u and v in residual graph
$C_f(u, v)$	residual capacity from node u to node v
$excess[u]$	excess flow of node u
$f_*^{max/min}$	max or min source flow in the flow network
$f_u(v, w)$	flow from node v to w with source node u

3.2 System Design

The proposed system design includes both network and MAC layer protocols. In the network layer (Section 3.2.1), a max-flow min-variance algorithm is proposed to maximize network throughput and fairness in the heterogeneous duty-cycle network. In the MAC layer (Section 3.2.2), with the time synchronization service from hardware, a localized TDMA MAC called TreeMAC [58] is propose to synchronizes nodes' duty-cycles in the "multi-path" flow network. A low-cost WWVB radio receiver is proposed to provide the accurate time synchronization. The complete hardware and software system has been implemented in both real sensor network testbed and simulators.

3.2.1 Network Algorithm for Throughput Maximization and Fairness

The software design involves both network and MAC layer. Subsection **??** and **??** describes our collaborative data delivery algorithm which maximizes network flow and fairness on heterogeneous duty-cycled network. In subsection 3.2.2, a localized TDMA MAC protocol is presented for duty-cycle synchronization to support the above collaborative data delivery protocol.

Loop-free Max-flow Algorithm for Throughput Maximization In this section, we present our "loop-free" maximum flow algorithm to collaboratively utilize heterogeneous duty-cycles for throughput maximization. Besides, various optimizations are also given in Algorithm 1, including eliminating the flow loop to improve the energy efficiency.

The distributed "Push-Relabel" algorithm in [59] solves the max-flow problem under the edge-capacited network. Initially, the source node has a valid height and start a "push" operation. "Relabel" operation adapts the height in each node, so that data flow seeps through the network just as water flow through a terrain. However, it can not directly apply to our problem due to two main reasons. First, the capacity constraint is not on edges, but on nodes. In our problem, the constraint is that the total flow amount of the incident edges of a node u can not exceed its capacity. As stated earlier, each node has a known capacity

constraint, while each edge's capacity is unknown and depends on its surrounding nodes. Second, there are multiple sources and single sink. All nodes in the network except the sink sense the environment and deliver data toward the sink, and they need to get a fair flow allocation.

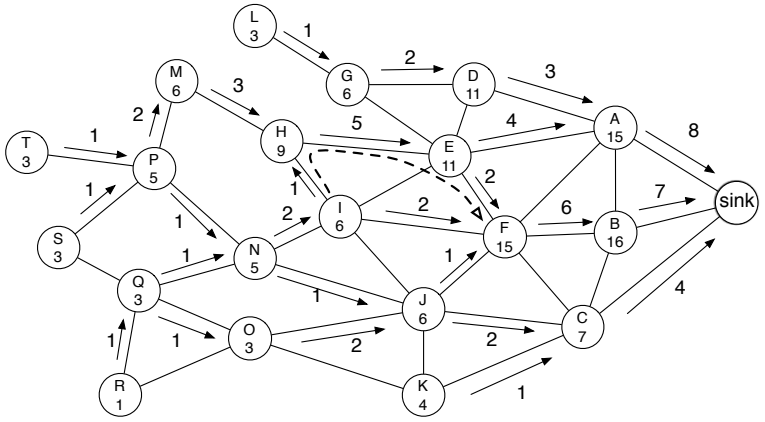


Figure 3.4 Max-flow network with long push loop

Besides the major differences, it is also worthy to point out the "Long Push Loop" problem, which hinders the efficient data delivery. It is not hard to see that the "PUSH" operation in original push-relabel algorithm chooses a valid path in an arbitrary way, which may result in loops. And we have found the "Long Push Loop" problem in the arbitrary "PUSH" operation, illustrated in Figure 3.4. In Figure 3.4, node I with capacity 6 has 2 inflows. Assuming it has four admissible leaving edges in the residual graph, (I, H), (I, J), (I, E) and (I, F), node I can push its excessive flow to any of these edges. If node I arbitrarily selects node H to push, the flow may go through H, E and back to F. Compared to pushing flow directly to node F, this arbitrary push may result in longer data delivery path. And if each node maintains the arbitrary selection for PUSH method, the long-loop path may exist.

In Algorithm 1, we present our loop-free push-relabel max-flow algorithm. At the beginning, the height of every node is initialized as 0. And a source rate D_u is assigned to every node except sink. Notice that, this step is the same as adding a virtual source connecting every source node with link capacity D_u. After that the algorithm executes the push-relabel

operation on each overflowed node to solve the max-flow algorithm. A key difference is that in line 4 of Algorithm 1, the $C_f(x, y)$ is determined by the minimum node residual capacity of node x and y: C_x and C_y. The reason is that one node may deplete its energy or reach its node capacity before the transmitting data rate exceeds the limit of its incident link. In the Algorithm 1, each node implements the PUSH operation with a careful selection on neighbors with residual capacity. The neighbors with the shortest path to sink and larger node degree will have higher priority. The rationale of this selection strategy is that the neighbor with the smallest hop-count and largest degree will likely yield more opportunities to deliver data to the sink. When the Algorithm 1 terminates, it generates a maximum flow assignment in the network, which can be easily converted to each individual node's maximum transceiving data rate (e.g., T_u, R_u) thereafter.

We now first show that the loops in flow path are eliminated by our algorithm.

Lemma 1 *In the flow network generated by Algorithm 1, each of its flow paths is loop-free.*

Proof 2 *The loop-free flow path can be proved by contradiction. Assuming there is a loop flow path $u, v, ..., w$, where the edge (u, w) also belongs to the original graph. The "PUSH" method given by Algorithm 1 can select v as the parent of u since the shortest path has highest priority. It implies that the flow path $u, v, ..., w, ..., sink$ is the shortest path. On the other hand, edge (u, w) is also selected due to the shortest path preference, which means these two shortest paths shall have equal length. We then assume that the length of path from w to the sink is l, the length of flow loop is the summation of l and length of $u, v, ..., w$. Contradiction is induced.*

To prove Algorithm 1 does generate maximum network flow when it terminates, we need to show that it keeps the height function definition (which is a key condition in Golberg-Tarjan's push-relabel algorithm [60]).

Lemma 3 *In Algorithm 1, the substitution of node capacity for link capacity still keeps function h as a height function.*

Algorithm 1 Loop-free Max-Flow Algorithm

procedure: *InitializeNetwork(V,E)*

1: **for** each vertex $x \in V \setminus sink$ **do**
2: add virtual link connecting x and virtual source
3: $excess[x] \leftarrow D_x$; $h[x] \leftarrow 0$
4: **end for**

procedure: *Push-Relabel(V,E)*

while ∃ x ∈ V \ *sink* && $excess[x] > 0$ **do**
2: $U_x \leftarrow \emptyset$; $W_x \leftarrow N_x$
while $W_x \neq \emptyset$ **do**
4: **if** $C_f(x, y) > 0$ **then**
$U_x \leftarrow U_x \bigcup y$
6: **end if**
$W_x \leftarrow W_x$ - y
8: **end while**
now U_x is the set of node x's neighbors with residual capacity. Sort U_x in ascending order of their hop-count to sink and descending order of node degree (when hop-count is same).
10: update the shortest path from x to *sink*
if $U_x \neq \emptyset$ **then**
12: **while** $excess[x] > 0$ && $U_x \neq \emptyset$ **do**
y ← $U_x[head]$
14: **if** $h[x] = h[y] + 1$ **then**
$d_f(x, y) \leftarrow min\{excess[x], C_f(x, y)\}$
16: Push($d_f(x, y)$, x, y)
excess[x] ← excess[x]-$d_f(x, y)$
18: **end if**
$U_x \leftarrow U_x$ - $U_x[head]$
20: **end while**
else
22: Relabel(x): h[x] ← 1+ $min\{h[y] : (x, y) \in E_f\}$
end if
24: **end while**

Proof 4 *In the step of $InitializeNetwork(V, E)$, the h is initialized as a height function. Therefore, the h will be valid if both $RELABEL$ and $PUSH$ methods leave h a valid height function. First, we look at the $RELABEL$ operation, which is executed in the overflowed node. If there is a residual edge (x,y) $\in E_f$ which leaves x, then after the relabel operation on node x, $h[x] \leq h[y] + 1$ follows. Since we use the node capacity, there are no residual edges entering to an overflowed node. Thus, the $RELABEL$ operation holds h as a valid height function. Second, $PUSH$ can only occur in the edge (x, y) with $h[x] = h[y]+1$, which results in a residual link (y, x) with $h[y] = h[x] - 1 \leq h[x] + 1$. Thus, the height function still holds after the $PUSH$ operation.*

Proven the height function retains as a valid function during Algorithm 1, we then prove the correctness of our algorithm. In other words, the maximum network flow can be achieved at the time when algorithm terminates.

Lemma 5 *When Algorithm 1 terminates, the computed preflow in the network is a maximum network flow.*

Proof 6 *When the algorithm terminates, there is no overflowed node in the $V \setminus sink$. So, the preflow becomes the actual network flow. The Lemma 3 proved that using node capacity instead of link capacity still retains h as a valid height function. Combining with the Lemma 26.18 in [61], we can conclude that there is no path from source to sink in the residual graph when algorithm terminates. Therefore, the preflow is a maximum flow according to the max-flow min-cut theorem.*

Besides the correctness of algorithm, we also find the upper bound for algorithm's time complexity.

Lemma 7 *(Upper bound of saturating pushes) Let $G = (V, E)$ be a flow network generated by Algorithm 1, the number of saturating pushes in Algorithm Push-Relabel(V,E) are less than $2|E|$.*

Proof 8 *The calculation of saturating pushes is associated with a given connection between node x and y. If the edge (x, y) is in the original flow graph G, a saturating push can occur only if $h[x] = h[y] + 1$. Since height function h can only increase, the next saturating push occurs in the edge (y, x), when $h[y] = h[x] + 1$. The second saturating push makes node y an invalid push neighbor of x, due to its empty node capacity. It implies that the number of any edge (x, y) in the original graph can have saturating push no more than twice. Multiplying by the number of edges in the graph renders a upper bound of less than $2|E|$ saturating pushes in total.*

Lemma 9 *(Upper bound of nonsaturating pushes) Let $G = (V, E)$ be a flow network generated by Algorithm 1, the number of nonsaturating pushes in Algorithm Push-Relabel(V,E) is bounded by $4|V|(|V|^2 + |E|)$.*

Proof 10 *According to the Lemma 26.24 in [61], we build a potential function $\delta = \sum_{v:e(v)>0} h[v]$. Initially, δ is equal to 0, and the value of δ is equal to 0 again when the algorithm terminates with no excess flow in the nodes. According to the Lemma 3, h is a valid height function, thus for all nodes $h[x] < 2|V|$ according to Lemma 26.21 in [61]. So, by Corollary 26.22 of [61], the number of relabel operations is bounded by $2|V|^2$. Thereafter, the total relabel operations increase the δ by at most $(2|V|)(2|V|^2)$. Also, according to Lemma 7, the number of saturating push operations is at most $2|E|$, so the δ can be increased by at most $(2|V|)(2|E|)$ during saturating pushes operations. Consequently, the δ is at most $4|V|(|V|^2 + |E|)$. The nonsaturating pushes will decrease the δ every time it is executed. If a nonsaturating push is from node u to v, node u will change from overflowed node to non-overflowed node, and node v will become overflowed when the nonsaturating push is more than its own source data rate. So, each nonsaturating push will decrease the δ by at least $h[u] - h[v] = 1$. Thus, the upper bound of nonsaturating pushes is $4|V|(|V|^2 + |E|)$.*

Consequently, we have the following theorem:

Theorem 11 *The Algorithm 1 generates a loop-free maximum network flow and terminates within $O(|V|^2|E|)$ operations.*

Proof 12 *As proved in the Lemma 3, function h is kept as a valid height function throughout the Algorithm 1. And there is no path from virtual source to the sink in the residual network G_f, which implies that the assigned flow in the network is the maximum network flow. Combining the Lemma* 26.22 *in [61], Lemma 7 and Lemma 9, the number of operations in the Algorithm 1 is up bounded by $O(|V|^2|E|)$. In other words, it terminates within $O(|V|^2|E|)$ operations.*

In Algorithm 1, we describe it in a centralized way for simplicity of presentation and understanding. It can be easily implemented as a distributed network protocol, just like the original push-relabel algorithm [59] can be implemented in distributed fashion.

Flow Balance Algorithm for Fairness Though Algorithm 1 maximizes network throughput in a duty-cycled network, it does not guarantee flow fairness. In Algorithm 2, we propose a flow balance algorithm, providing fair flow assignment among nodes while maintaining maximum network flow. Unlike some other existing works which rely on the predetermined tree structure, our min-variance flow balance algorithm can apply to general data collection structure.

In order to apply the distributed flow balance algorithm, each node maintains two flow vector: $F_u(f_{min}, f_{max}, \Delta f)$, which is the accumulated min flow, max flow and the adjustable flow amount; $F_u^{'}(f_1, f_2, ..., f_k)$, which are the flows from its direct neighbors. The overall idea of distributed Algorithm 2 is to iteratively select the f_u^{max} and f_u^{min} flows out of flow set, and try to find the shortest augmenting path between them. In each iteration of $MinVariance$ of Algorithm 2, the routine $FSAP(f_x^{min}, f_x^{max}, G)$ will find a shortest augmenting path with either positive or zero adjustable flow amount. If an augmenting path is found in $FSAP()$ method, the $FSAP()$ returns a positive Δ_f. Then these two flows can be balanced by letting f_u^{max} yield Δ_f amount of flows to f_u^{min}. This is described in line 6 to 8 of procedure: $MinVariance$. After the first pair of f_u^{max} and f_u^{min} has been solved, we rearrange the

Algorithm 2 Distributed Min-Variance Flow Balance

procedure: *MinVariance(u)*

while $(f_{min} \neq f_{max})$ *and* $\Delta f_{min} > 0$ **do**
  $F_u \leftarrow \{f_{min}, f_{max}, \Delta f_{min}\}$
  $F_u^{'} \leftarrow \{f_1, f_2, ..., f_k\}$
  $f_u^{max} \leftarrow f_{max}$; $f_u^{min} \leftarrow f_{min}$
  $\Delta f = FSAP(f_u^{min}, f_u^{max})$
  if $\Delta = \max\{\Delta f, \Delta f_{min}\} > 0$ **then**
    $f_u^{min} \leftarrow f_u^{min} + \Delta$; $f_u^{max} \leftarrow f_u^{max} - \Delta$
    Update Δf_{min} along the path
  else
    Find new f_{min} by BFS and update Δf_{min}
  end if
end while

procedure: *FSAP*($f_x^{min}(*,*)$,$f_y^{max}(*,*)$)

$U^{'} \leftarrow x$; $P \leftarrow \emptyset$; $\Delta f \leftarrow 0$
while $\delta_f(x, y) < |V|$ **do**
  if G_f contains an admissible edge $(U^{'}, U^{''}) \in E_f$ **then**
    let $(U^{'}, U^{''})$ be an admissible edge in E_f
    $\pi(U^{''}) \leftarrow U^{'}$; $U^{'} \leftarrow U^{''}$
  end if
  if $U^{'} = y$ **then**
    use π to find an augmenting path P
    $\Delta f \leftarrow \min((f_y^{max}(*,*) - f_x^{min}(*,*))/2, C_f(P))$
    return Δf
  else
    Relabel($U^{'}$) ; $U^{'} \leftarrow \pi(U^{'})$
  end if
end while
return Δf

flow set accordingly, and repeat the procedure until there is no more adjustable flows in the network. Otherwise, we will update the f_{min} in line 10 by removing the unadjustable f_{min} from corresponding node, i.e. node x, and recompute the f_{min} for node x based on its F'.

The time complexity for $FSAP()$ is $O(|E|)$, and it have at most $|V|$ iterations. And the algorithm will terminate when f_{min} equals to the f_{max}, which is bounded by size of vertex set $|V|$. Therefore, the complexity of Min-Variance Flow Balance Algorithm is $O(|V|^2|E|)$.

3.2.2 Localized TDMA MAC Protocol for Duty-cycle Synchronization

To actually achieve the performance of aforementioned algorithm, duty-cycle need to be synchronized in MAC layer. We proposed a localized TDMA MAC protocol with the objective of synchronizing duty-cycles on the selected network flow paths and eliminating media access contentions. In order to support the time synchronization in TDMA MAC protocol, we design a low-power hardware by utilizing the WWVB atomic clock receiver. We select the CME 6005 [62] as the receiver chip, which is of only about 10 dollars, in our hardware design. Its current consumption is less than $100\mu A$ in full active mode with the voltage supply of 3.0V, which is much lower energy cost than software-based time synchronization protocols, like FTSP [63]. The software-based time synchronizations need to turn radio on and exchange time messages periodically so that clock drift of sensor nodes can be compensated.

The MAC protocol is inspired from TreeMAC [58] - a localized TDMA MAC protocol. The main idea is to utilize parent-child relationship in a data collection tree to minimize the signaling overhead. A time *cycle* is divided into *frames* and frame into *slots*. The parent determines each children's *frame* assignment based on their relative bandwidth demand, and each node calculates its own *slot* assignment based on its hop-count to the sink. The control message is between parent and child only. This simple yet effective 2-dimensional frame-slot assignment algorithm can avoid schedule conflicts in network flow of shortest path tree.

However, TreeMAC [58] can not directly apply to the "multi-path" flow network, as flow network topology is not a tree structure. Fortunately, such a many-to-one flow network graph

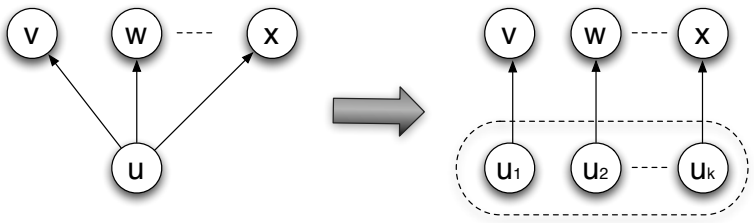


Figure 3.5 Virtualization example.

can be virtualized as a tree structure. Basically, a node u with k parent can be virtualized as k virtual nodes $\{u_1, u_2, \cdots, u_k\}$, and then each u_i $(1 \leq i \leq k)$ now have a single parent, as illustrated in the figure 3.5. This virtualization can be further extended: for every node u in the network (even if it has a single parent or single path to the sink), assuming it has m-unit flows to forward, it can also be virtualized as m virtual nodes $\{u_1, u_2, \cdots, u_m\}$. Then each link with m flow units can be virtualized accordingly and a virtual tree structure emerges.

The frame-slot assignment algorithm of TreeMAC can apply to the virtual tree structure now. For slot assignment, we can directly apply the slot assignment algorithm mentioned above. For frame assignment, assume each flow has a flow ID f_{ID}, we can assign all nodes on the flow path with the f_{ID}-th frame. Now, different flows are guaranteed to be conflict-free, since they have different frames per cycle. However, this will require that the time cycle size is bigger than the number of flows. This is not a problem, because the number of flows is constrained by the sink node's capacity.

Another concern on the difference between tree and virtual tree is: in a tree structure, every node only has one parent to forward data, and its hop-count can be determined unambiguously; but in a flow network, a node u may have a different hop-count through different selected flow paths: $(u, v, ..., y, sink)$ and $(u, w, ..., y, sink)$. Thus the virtual nodes u_i and u_j from the same node u may have a different hop-count too. Fortunately, this is also not a problem. This just means that the original node u will get different slots in different frames, instead of the same slot in different frames.

Theorem 13 *The duty-cycle synchronized MAC protocol based on TreeMAC ensures conflict-*

free transceiving schedules in maximum flow networks.

Proof 14 *Using TreeMAC frame-slot assignment in the virtual tree, the frames assigned to different virtual nodes are non-overlapping, if those nodes do not have ancestor-descent relationship. Thus, there must exist no schedule conflict between different unit-flow paths. In addition, according to Lemma 1, our flow paths have no loops. Since parent and child have no schedule conflicts in TreeMAC, any two nodes in the same flow path also do not have schedule conflicts.*

3.3 System Evaluations

3.3.1 Experimental Evaluations

In this section, we evaluate our system design and implementation, including hardware design, MaxFairFlow algorithm (Loop-free Max-flow and Min-variance fairness algorithm) and localized TDMA, by comparing to the commonly used TinyOS protocol stack in three evaluation criteria: *Data Throughput*, *Data Delivery Latency* and *Energy Efficiency*.

We show our experimental testbed in Figure 3.6, topology in Figure 3.7. The WWVB antenna is mounted at the top of the pole, and TelosW [64] is attached to the WWVB receiver circuit board. We implement and deploy our energy synchronized system in the sensor network testbed of 15 TelosW nodes, which are placed in a square 10*10 meters area with multihop topology. The sink node is placed in the upper left corner.

In Figure 3.7, each node is indicated by a tuple of ID and node capacity. Low radio transmission power is configured to form a multihop duty-cycled network, with maximum 5 hops.

Data Throughput We analyze the data throughput to validate the data collection efficiency of our protocol. Figure 3.8 shows the comparison of the data throughput between the CTP-XMAC and MaxFairFlow-TDMA algorithm. With the total sending packet number of 3600 in each node, the minimum node throughput for CTP-XMAC is $2,916$, which is approximately 81.0% in terms of packet delivery ratio. It happens on node 13 with hop-count

Figure 3.6 Outdoor experiment test-bed of 15 nodes synchronized by WWVB radio.

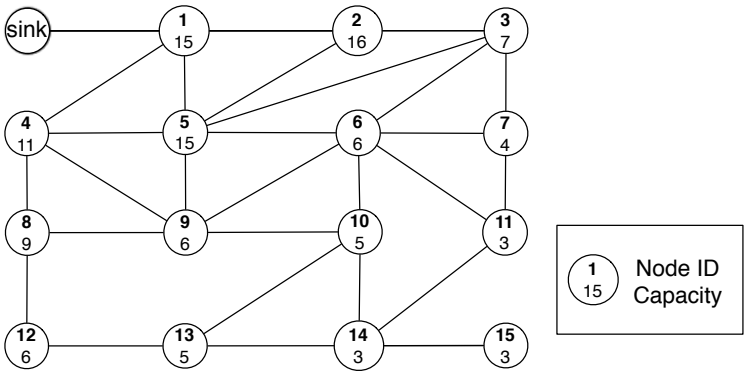


Figure 3.7 The snapshot of topology of testbed. The upper number inside the circle denotes the node ID, the lower one is the node capacity.

4. And the maximum throughput is 3,519 on node 1. In the network, the total throughput is 6.6% lower than that of MaxFairFlow-TDMA.

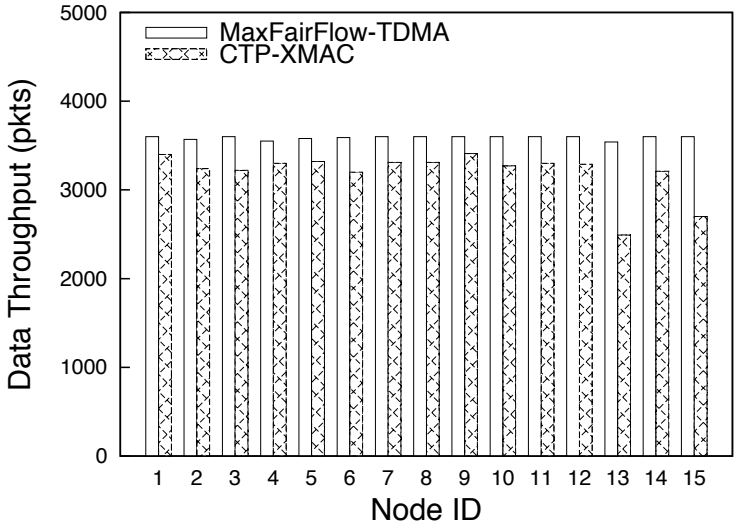


Figure 3.8 Data throughput.

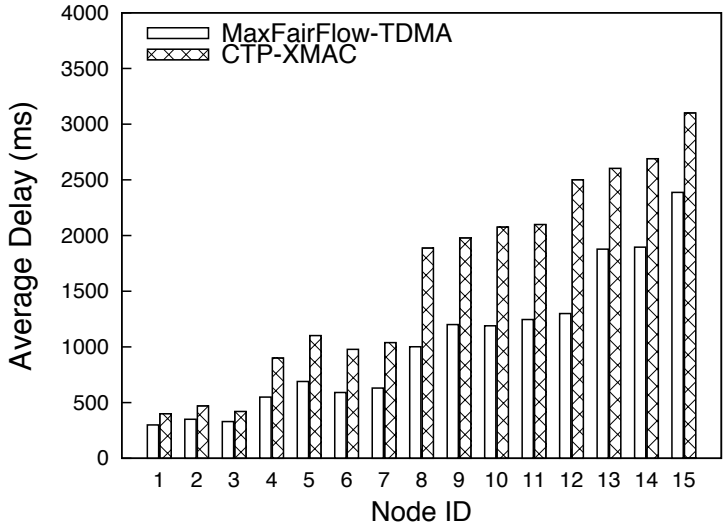


Figure 3.9 Average delivery delay in the network.

The reason for the higher packet delivery ratio is that MaxFairFlow-TDMA algorithm takes advantages of the "multi-path" for collaborative data forwarding. The max-flow and min-variance algorithm can achieve the optimal forwarding paths for each node, bypassing

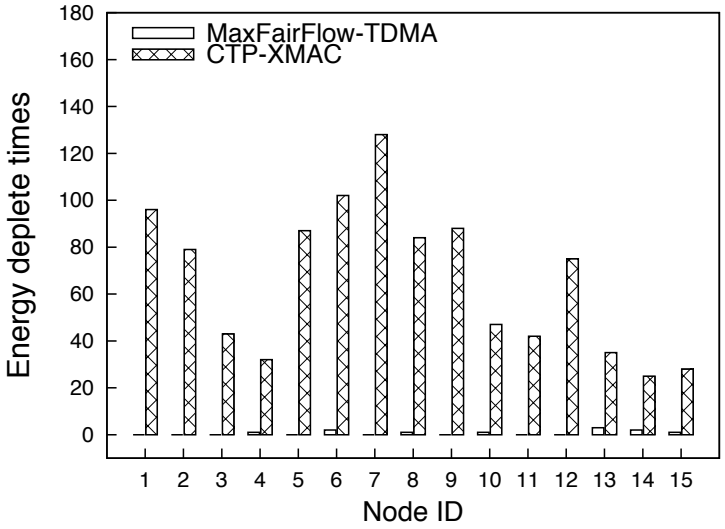


Figure 3.10 The number of times of depleting quota energy.

the congested node. For each node, our algorithm can increase the forwarding probability of each packet, while for the entire network, the data traffic is balanced over different delivery paths.

Data Delivery Latency The low-duty-cycled network poses a critical challenge on the timeliness of delivered data. We have evaluated the data delivery latency in the network against multiple hopcounts. Figure 3.9 shows that the latency of CTP-XMAC increases proportionally to the hopcount, while the rate of increasing latency in our algorithm is much slower. It also shows that the maximum latency for CTP-XMAC is 3100 ms, which is 5 hops away from sink. But the maximum for our MaxFairFlow-TDMA protocol is only 2387 ms at node 15. It is because that in MaxFairFlow-TDMA, nodes wake up in the designated slots with synchronized time, without any delay between the wake-up time of sender and receiver. Overall, data delivery in TDMA with assigned synchronized slots can provides better latency for QoS required applications.

Energy Efficiency The energy efficiency is evaluated in this section. Instead of estimating energy from communication packet number or attaching the oscilloscope physically to the mote, we exploit the on-board energy meter to measure the real-time energy consumption rate [64]. The benefit is that it evaluates all the real energy consumption including RF

transceivers and WWVB radio.

Figure 3.10 shows the total number of times nodes deplete their quota energy within experiment. Respecting the planned lifetime, each node has a derived energy cap in each time frame. With the realtime energy consumption measurement from energy meter, a node can decide if it runs out of the derived "quota energy". In Figure 3.10, the maximum energy deplete times for CTP-XMAC approach is 128 times, and every node has, at least 30 times, depleted its own energy. The Figure 3.10 shows that the proposed algorithm almost never deplete the quota energy, less than 5 times in node $6, 13$ and 14 respectively. It is clear that different duty cycles and asynchronous wake-up times give the asynchronous scheduling protocol (e.g. CTP-XMAC) a challenge, which can cost more energy in holding on the radio and waiting for the receiver to wake up. The reasons for the overall advantage of energy consumption gained in MaxFairFlow-TDMA are two-folded: (i) wake-up schedules are synchronized among different duty-cycled nodes; (ii) energy consumption is more balanced through collaborative multi-path selection in the Max-Flow Algorithm and Min-Variance Flow Balance Algorithm.

Moreover, we examine the network lifetime under different protocols. Network lifetime is the network up-time from boot-up to the time when the first node dies. In Figure 3.11, the first node goes down after 180 hours in CTP-XMAC, and the maximum lifetime for nodes in the network is 221 hours. In MaxFairFlow-TDMA, nodes were running 231 hours before the first node depletes its residual energy, which is even longer than the maximum node uptime in CTP-XMAC. Extended network lifetime is due to that our MaxFairFlow-TDMA algorithm balances the network traffic through multiple paths, and deliver the sensing data in a collaborative way. In Flow Balance Algorithm, the assigned flows are balanced in each flow path, achieving maxmin and minmax fairness. Consequently, the algorithm balances the energy loads in multiple paths as well. All of these contribute to improving the network lifetime.

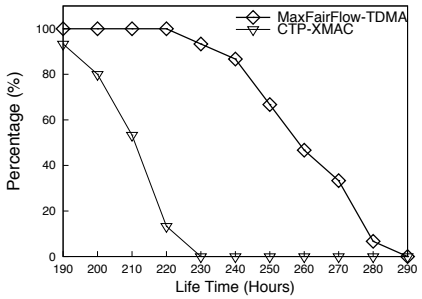


Figure 3.11 The CDF of expected life time in the network.

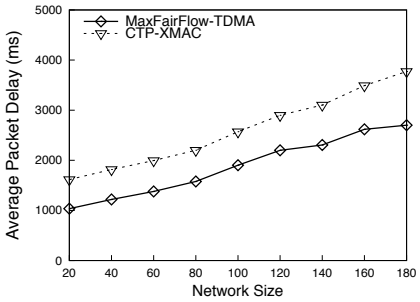


Figure 3.12 Maximum and Average Delay in different network sizes.

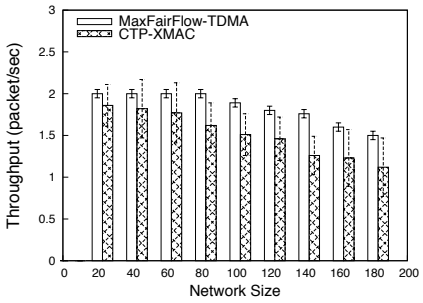


Figure 3.13 Average throughput per node against different network sizes.

3.3.2 Simulation Evaluations

To show the scalability of our proposed algorithm, we have evaluated the performance on a larger scale with TOSSIM. All the sensor nodes are placed in a 100*100 square grid, with sink node on the left upper corner.

Network Delay The average and maximum packet delay is evaluated with network size ranging from 20 to 180. And the maximum delay is affected by the maximum hopcount in the network. This is why the maximum delay increases from 3100 ms of 20-node network to 20000 ms in a 180-node network in Figure 3.12. And the delay of MaxFairFlow-TDMA scheduling has a much slower increasing slope, ending up with the 12000 ms delay in the network with 180 nodes.

In terms of average delay over the different network sizes, both of the two protocols have the slow increasing trend. This is because the average hopcount does not increase sharply as the network size goes larger. It is important to mention that the MaxFairFlow-TDMA

method still gets an advantage of 25.2% in the average packet delay. That is because that collaborative multi-path selection makes the distribution of nodes more balanced, resulting in a smaller average hop count.

Network Throughput We have measured the average throughput per node in different network sizes. In Figure 3.13, the bar denotes the average throughput for each node of a given network size, while the error reflects node throughput deviation in that network size. As the network size scales, in CTP-XMAC, it is observed that the throughput per node starts to decrease at the size of 40, and continue declining to the 1.12 packets / second in each node, at the network size of 180. For the MaxFairFlow-TDMA, the network starts to saturate at the size of 80, and the throughput for each node decreases until 1.5 packets per second. Moreover, it is observed that the variance of throughput for each node remains within the value of 0.05 packets per second in MaxFairFlow-TDMA. However, the variance for CTP-XMAC is much higher, in which the maximum variance is 0.36 packets per second, with average variance of 0.30. It is because the Flow Balance Algorithm of section **??** can efficiently even out the traffic flows and take advantages of multi-path for delivering data packets.

CHAPTER 4

DATA COMMUNICATION WITH ENERGY-FREE NETWORK CODING (ONEC)

The erasure codes were initially proposed to enable reliable broadcast over "one-to-many" communication conditions, where feedback is very expensive or even infeasible, such as satellite broadcast or Internet video broadcast applications. LT (Luby Transform) code [17] is the first effective implementation of erasure codes with low encoding and decoding complexity. Due to its low complexity in both encoding and decoding sides, it is appropriate to be utilized in wireless sensor nodes where the computing capability is limited. However, the redundant coding overhead introduced can compromise its efficiency, if LT code applies without alteration to the network where multiple data sources exist. The core of LT code is the RSD (Robust Soliton Distribution), which is a probability distribution of encoding degree for each packet. The data recovery can be proved as high probability, providing that each degree in encoding packet is independently drawn from RSD, and sufficient amount of packets arrive at decoder, e.g. $K+c\cdot\sqrt{K}\cdot\ln^2(\frac{K}{\delta})$, and K is the symbol size. As each node encodes its own k symbols using individual RSD, the overhead for successfully decoding, $N\cdot c\cdot\sqrt{k}\cdot\ln^2(\frac{k}{\delta})$, scales with the network size N. In the other hand, if we view all the source symbols as originating from single "super node", allowing recoding in the forwarding nodes, the message overhead will be significantly reduced to $c\cdot\sqrt{N\cdot k}\cdot\ln^2(\frac{N\cdot k}{\delta})$ for large network size N. For example, in Figure 7.1, every node has different symbol sizes to send, e.g. 50 or 100 respectively, totaling 400 in the network. In the first case shown in the upper graph of Figure 7.1, individual RSD is applied to generate encoded packets, and no recoding occurs in the forwarding nodes. The total amount 650 of packets are required for high-probability recoding, where message overhead $R=\sum_i c\cdot\sqrt{k_i}\cdot\ln^2(\frac{k_i}{\delta})=4\cdot 0.1\cdot\sqrt{50}\cdot\ln^2(\frac{50}{0.05})+2\cdot 0.1\cdot\sqrt{100}\cdot\ln^2(\frac{100}{0.05})=250$. However, in the lower graph, each node derives a degree distribution from RSD, and encodes

packet accordingly. Recoding in forwarder nodes make the final degree distribution conform to the RSD, with much less redundant overhead. The message overhead R can be calculated as: $R = c \cdot \sqrt{\sum_i k_i} \cdot \ln^2(\frac{\sum_i k_i}{\delta}) = 0.1 \cdot \sqrt{400} \cdot \ln^2(\frac{400}{0.05}) = 162$. The total packets entailed by successful recoding is only $400 + 162 = 562$, which is 13.7% less under the total symbol size of 400.

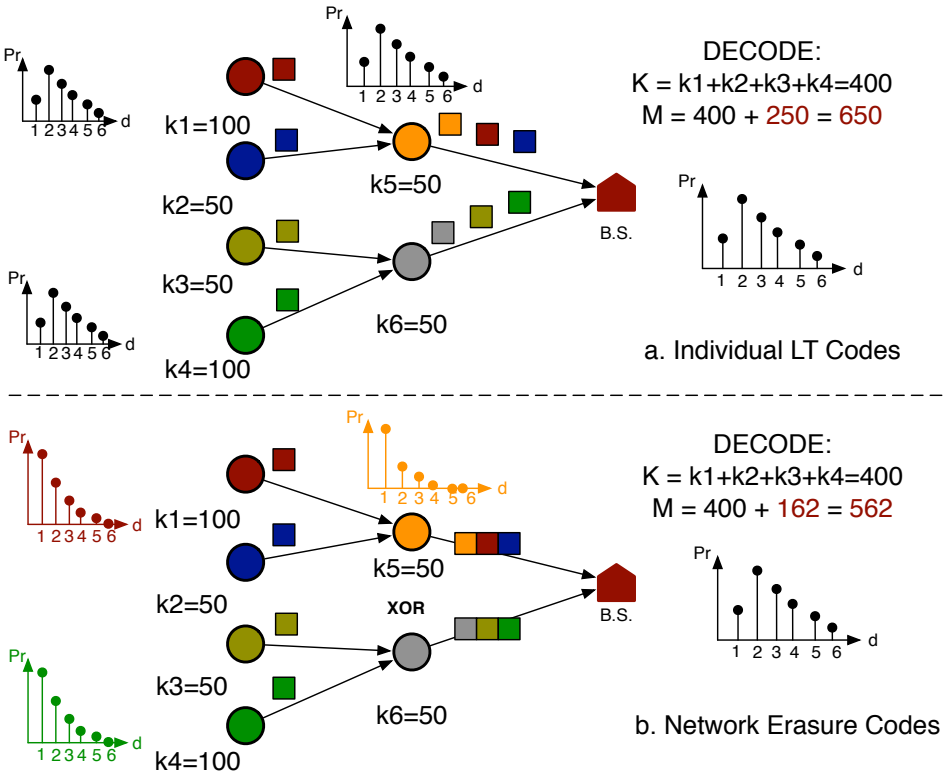


Figure 4.1 The Comparison of network erasure code and individual LT code in network.

Moreover, intermediate recoding takes advantage of opportunistic overhearing in wireless broadcasting to combat communication loss. Those two reasons motivate the design of ONEC for a collaborative data collection.

In this paper, we present the design, implementation, and evaluation of *ONEC*, an Opportunistic Network Erasure Coding protocol for reliable data delivery over disruptive sensor networks. ONEC is designed to transform the "end-to-end" paradigm of LT codes to "many-to-one" network LT codes. Firstly, it adopts recursive decomposition structure to conduct probability distribution deconvolution, supporting heterogeneous symbol sizes

in each source nodes. Secondly, it allows every node to carry out selective recoding from its own sensing data and received packets. Thirdly, it enables opportunistic recoding on the overheard packets, based on relay hop distance. It can compensate the information deficit due to lossy transceiving between neighbor nodes. Lastly, ONEC make packet degree distribution closely approximate the RSD, which enables high probability decoding with low computation complexity. The preliminary work is published in [36], and we provide comprehensive design and theoretic analysis details in this article.

The main contributions of ONEC protocol are as follows:

- *Low coding overhead*: ONEC utilizes the intermediate recoding to reduce the redundant encoding overhead. The recursive deconvolution only takes place during network initialization and the time when network dynamics happen. The control message cost for relaying decomposed degree distribution message is minimized.

- *Disruption tolerance*: By enabling opportunistic encoding in the intermediate nodes on the forwarding paths, ONEC becomes resilient to disruptive networks. Opportunistic overhearing of symbols are incorporated into the encoded packets to make the upstream nodes sustainable in presence of some failure in downstream nodes.

- *Low latency and buffer space*: In ONEC, the intermediate nodes recode packets but not decode them. Hence, little buffer space is required to store and conduct XOR on received encoded packets.

- *Performance with low coding complexity*: Extensive performance analysis reveals the advantages of ONEC over existing methods, in terms of network goodput, message complexity and buffer space.

4.1 ONEC Protocol Design

In this section, we present the ONEC design, followed by detailed discussions of each component in ONEC. Illustrated in Figure 6.1, ONEC protocol is comprised of four major

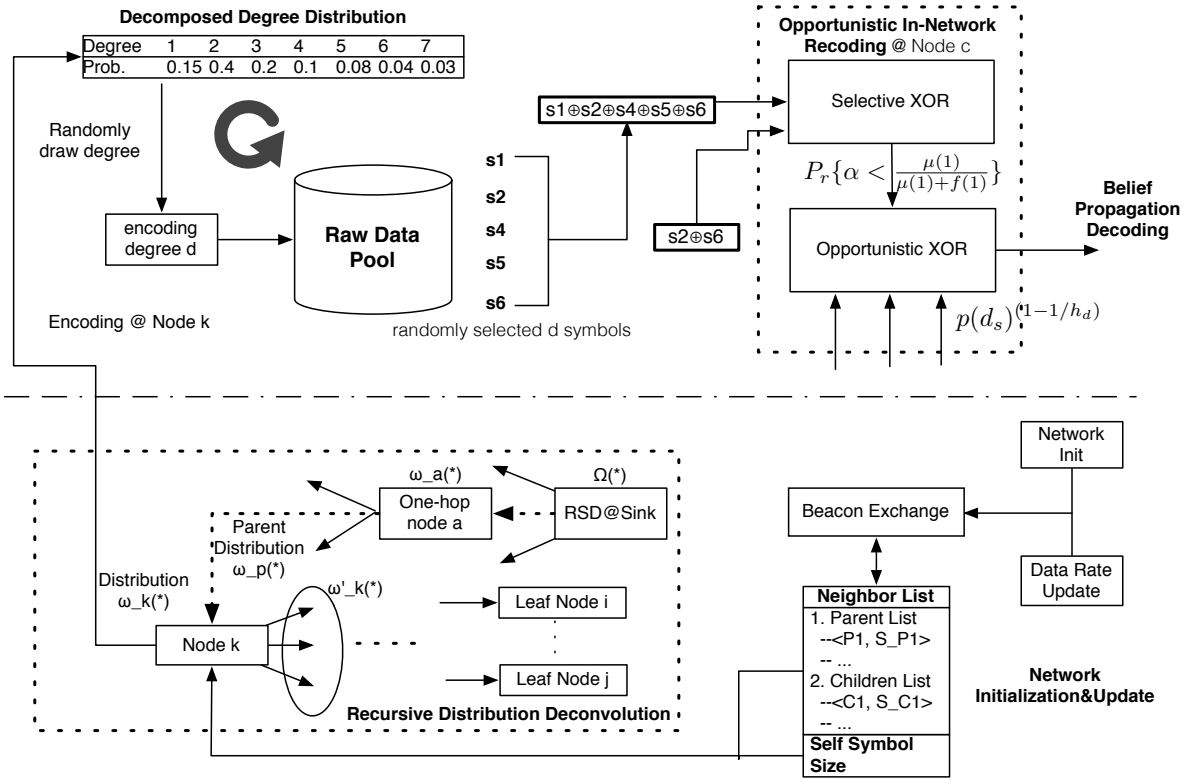


Figure 4.2 Overview of ONEC working flow.

phases: 1) *Network Initialization and Update*, in which nodes in network use broadcast beacon to acquire neighborhood information; 2) *Recursive Degree Deconvolution*, in which nodes conduct deconvolution of RSD in an recursive way; 3) *Opportunistic In-Network Recoding*, in which sensing nodes produce encoded packet based on derived degree distribution and forwarding nodes selectively incorporate the overheard symbols to the encoded packets; 4) *Data Decoding*, the decoder apply Belief Propagation algorithm to decode data with low computation complexity.

4.1.1 Network Initialization and Update

In our network model, continuous sensing data is divided into blocks or chucks, on which the erasure encoding and decoding are applied. Hence, the data rate R_i also denotes the total symbol size S_i for a specific time frame $\{t_i, t_i + \tau\}$. We use these two terms interchangeably in the following presentation. As a distributed protocol, nodes can derive

the degree distribution for itself, with only information of neighbors' status. Illustrated in Figure 6.1, a local table is established as a prerequisite for distribution deconvolution, including neighbor list, a tuple of $< P_i, S\_P_i >$ (parent degree distribution, parent symbol size), multiple tuples of $< C_i, S\_C_i >$ (children degree distribution, children symbol size), and the symbol size for itself. Parent-children relationship can be obtained by constructing a tree structure $T(V, E^{'})$ rooted at sink, where $E^{'} \subset E$, given a network $G(V, E)$ by an effective routing protocol. As network starts, Ci equals to $null$, since the distribution deconvolution initiates from root node, and then spread out to the whole network.

Without loss of generality, each node acts as a source node, which generates data stream at a specific rate. The source data rates are considered as heterogeneous across the network, as well as dynamic over the time. Here, we assume that data rate of each node is not changing so dramatically fast, and kept steady for a time stretch. Thus, the exchange and update beacons are suppressed when there is no variation among neighbors after network initialization. Once the topology or the sampling data rate alters, the beacon exchange and message update is launched to keep the neighbor table and degree distribution updated. The details of updating degree distribution upon the dynamic conditions is discussed in Section 4.1.2.

4.1.2 Recursive Degree Deconvolution

Degree Distribution of LT Codes For sake of self-contained explanation, we briefly review the general principle of LT codes before delving into the deconvolution details. LT codes was designed and proposed in [17]. Assuming k symbols in the source, encoder in LT codes first select a degree from the designed degree distribution, i.e., d, then XOR d randomly selected raw symbols to an encoded packet. The encoding process can be conducted in a rateless fashion, pushing out encoded packets continuously. With LT codes, decoder can recover k original raw symbols as long as receiving $K = k + O(\sqrt{k} \cdot ln^2(k/\delta))$ encoded packets with low computation complexity w.h.p. (with high probability). Note that during the decoding process, a set of covered but not yet processed symbols, called *ripple*, is critical to the success

of decoding, since decoding stops once the ripple is empty. Therefore, it is vital to keep the ripple size always larger than 0 during decoding process. LT codes [17] present its core: *degree distribution*. The *Ideal Soliton Distribution* is to guarantee the ripple size equals to 1 during decoding, while the *Robust Soliton Distribution* try to increase the ripple size large enough to survive the practical fluctuation in the course of decoding. We quote the definitions of these two degree distributions here, and refer the detailed description to [17].

Definition 1 Ideal Soliton Distribution (ISD) *[17]:*

$$\rho(i) = \begin{cases} 1/k, & \text{if } i = 1 \\ \frac{1}{i(i-1)}, & \text{if } 2 \leq i \leq k \end{cases}$$

Definition 2 Robust Soliton Distribution (RSD) *[17]:*

$$\mu(i) = \frac{\rho(i) + \tau(i)}{\sum_{i=1}^{k}(\rho(i) + \tau(i))}, 1 \leq i \leq k,$$

where

$$\tau(i) = \begin{cases} R/ik, & \text{if } 1 \leq i \leq \frac{k}{R} - 1, \\ R \cdot ln(S/\delta)/k, & \text{if } i = \frac{k}{R}, \\ 0, & \text{otherwise.} \end{cases}$$

with

$$R = c \cdot \sqrt{k} \cdot ln(\frac{k}{\delta})$$

Deconvolution Overview Since LT codes originate from the context of single source and destination communication, it introduces considerable redundant messages when applied to the network data collection without changes. Thus, to apply the LT codes to in-network erasure codes, which enables the recoding of encoded packet, we convert the desired *RSD* into individual degree distributions for each node based on its own symbol size. Though we employ the similar idea of deconvolution as in [31], our approach can adapt to different

network topology with lossy or lossless link connection. Moreover, the bound for expected encoding symbol under recursive deconvolution is proved.

The overall idea of the deconvolution is that the process is initialized from root, and conducted recursively level by level along the tree structure. Each parent computes and disseminates the deconvolved distribution for its children, the process repeats until the leaf node receives its distribution. Our differences are three-folded: *first*, the deconvolution method can support arbitrary number of children with different input symbol sizes, which is amenable to various topology. *Second*, we do not restrict the deconvolution of distribution function to the relay model network. Instead, we deconvolved the distribution function with the help of a converge-cast tree structure. Each layer derives its own degree distribution from its parent distribution respectively. In fact, the converged collection tree is only utilized for message dissemination, in section 4.1.3 we show that opportunistic recoding does not rely on particular tree structure. *Third*, this process is able to apply in a network with lossy link. If the deconvolution message lost, the receiving node can compute the distribution locally with information overheard from its other neighbors. The trade-off is that accuracy of locally deconvolved distribution is less than the distribution computed from parent.

Deconvolution Details As described in [31], the "spike" of degree distribution $\mu(i)$ needs to be removed before the recursive deconvolution can be applied. After the preprocessing the "spike" at $\mu(1)$, the distribution deconvolution uses "enumerative combinatorics". The essential idea is to explore all the combinations of different children degrees which can contribute to the parent's distribution. The deconvolved function $f(i)$ from $\mu(i)$ is given by [31]:

$$f(i) = \begin{cases} \sqrt{\mu(2)}, & \text{i} = 1 \\ \frac{\mu(i+1) - \sum_{j=2}^{i-1} f(j) f(i+1-j)}{2 \cdot f(1)}, & 2 \leq \text{i} \leq \frac{k}{R} \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

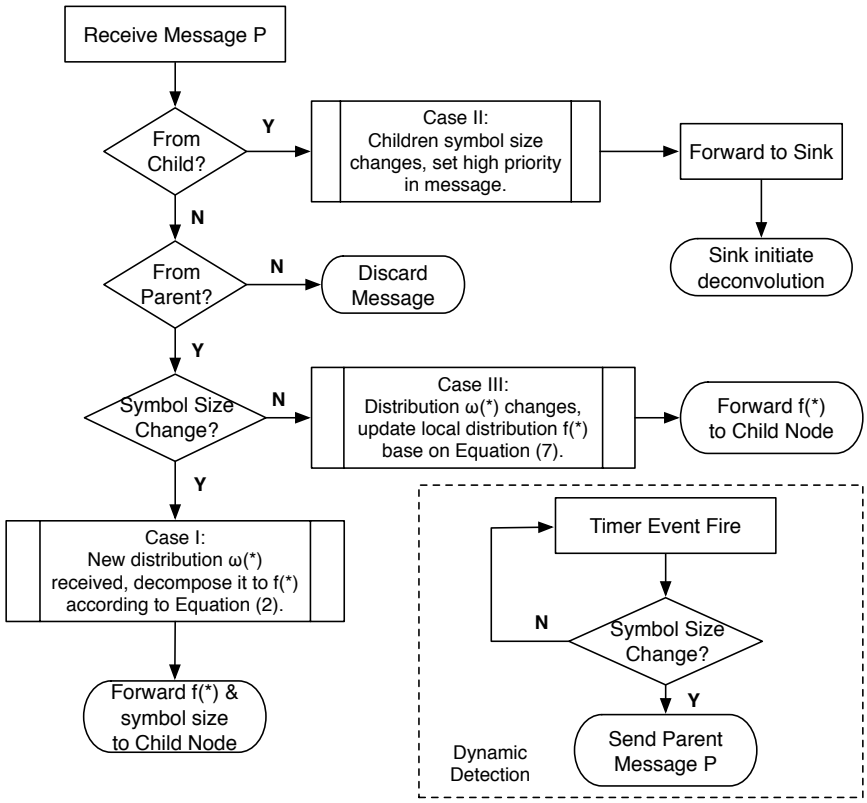


Figure 4.3 Flow chart for recursive degree distribution deconvolution.

We extend this basic deconvolution to apply in any tree structure, by clustering all the children into 2 child nodes (1 and 2) of different symbol sizes, say k_1 and k_2.

$$f(i) = \begin{cases} \sqrt{\mu(2)}, & \text{i} = 1 \\ \frac{\mu(i+1) - \sum_{j=2}^{i-1} f(j) f(i+1-j)}{2 \cdot f(1)}, & 2 \leq \text{i} \leq \Theta \\ \frac{\mu(i+1) - \sum_{j=2}^{\Theta} f(j) f(i+1-j)}{f(1)}, & \Theta < \text{i} \leq \frac{k}{R} \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

where $\Theta = min\{\frac{k}{R}, min(k_1, k_2)\}$, and i is the symbol index ranging from 1 to k_1 and k_2 respectively. We then show how to deconvolve the "spikes" of degree distribution $\mu(i)$, such that the recovered degree distribution from those two children conform with the parent's distribution. The deconvolved degree distributions given by Equation (1) does not deal with the "spike" points. The idea is to explore the individual probability that degree 1 symbol

falls into each of two data sets: k_1 and k_2. The probability of $\mu(1)$ split into two parts is proportional to the symbol size k_1 and k_2. Thus, the final deconvolved degree distribution is given as:

$$p_j(i) = \begin{cases} f(1) + \frac{k_j}{\sum_{j=1}^{2} k_j}\mu(1), & \text{j = 1, 2 and i=1} \\ f(i), & \text{i > 1} \end{cases} \tag{4.3}$$

Recursive Deconvolution. We relax the previous assumption that any subtree of a node consists of 2 child nodes (u and v). In other words, we make the above degree decomposition not only fit for the 2-child case, but also for more general m-child sub-tree case, in which m is not necessarily equal to power of 2. Illustrated in Figure 4.4, if a parent T has multiple children $A, B_1, B_2, ..., B_n$, the above combinatorics approach can not directly apply. Fortunately, we can do the node clustering and hierarchically deconvolve the distribution into each node. The recursive deconvolution in single level is illustrated in the right figure, which shows that all B_i node can first be considered as a *super* node, which has the total input symbol equal to $\sum_{i=1}^{n}(B_i)$. By this transformation, "2-child" deconvolution can be recursively conducted to generate the corresponding distribution function for each source node.

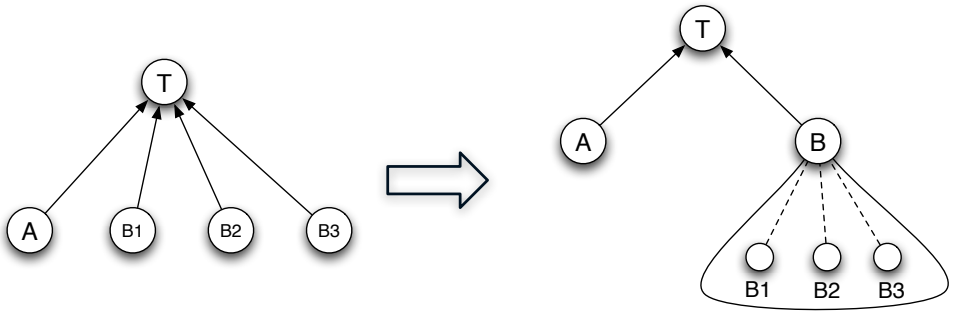


Figure 4.4 Illustration of recursive degree distribution deconvolution.

Selective Recovery. For recovering the "spike" ($i = 1$), we adopt selectively XOR to produce encoded packet, upon receiving the packet of "spike" degrees. In the receiver, the

recoding accept a packet of degree $i = 1$ with the probability: $P_r\{\alpha < \frac{\mu(1)}{\mu(1)+f(1)}\}$, where α is a random variable uniformly distributed within range $[0, 1]$.

Local Deconvolution. Practically in lossy link communication, the message for disseminating the computed degree distribution may get lost. Nodes which miss the dissemination message from its parent can still obtain degree distribution by conducting local computation. It yields a rough degree distribution by decomposing RSD recursively into two parts with $k - w$ and w symbols, assuming that node has w symbols.

Update Distribution. In dynamic network conditions, target distribution $\mu(i)$ for deconvolution operation can vary due to two reasons: first, topology alteration. Children nodes switch their parents, resulting in symbol size changes in subtree. Second, data sampling rate fluctuation. As illustrated in Fig 4.3, there are three update cases which address the dynamic distribution calculation and update.

Case I: Receive new distribution through initialization. Decompose distribution according to Equation (2).

Case II: Update on the variation of children symbol size. When one node receives the children symbol size update packet, it forwards this message to sink. Then, sink initiates the distribution deconvolution from RSD.

Case III: Update only on the distribution $\mu(i)$ from parent. Node and its children can apply the following equation to update their degree distribution without re-decomposing $\mu^{'}(i)$:

$$f^{'}(i) = \begin{cases} \sqrt{\mu^{'}(2)}, & \text{i} = 1 \\ f(i) + \frac{\mu^{'}(i+1)-\mu(i+1)}{2 \cdot f(1)}, & 2 \leq \text{i} \leq \Theta \\ f(i) + \frac{\mu^{'}(i+1)-\mu(i+1)}{f(1)}, & \Theta < \text{i} \leq \frac{k}{R} \\ f(i), & \text{otherwise} \end{cases} \tag{4.4}$$

where $f^{'}(i)$ can be derived by simple addition operation based on previous probability distribution $f(i)$.

Analysis of distribution deconvolution The degree distribution deconvolution closely approximate the desired degree distribution from the decoder perspective. Besides the effectiveness of distribution reconstruction, it is important to analyze the efficiency of proposed distribution deconvolution, in terms of message complexity and decoding robustness.

Message Complexity. We first give a message complexity bound on the data delivery based on distribution deconvolution approach.

Lemma 15 *Given k symbols, and distribution function $\mu(i)$ ($1 \leq i \leq \frac{k}{R}$)), the deconvolved function $f(i)$ satisfies:*

$$\sum_{i=1}^{\frac{k}{R}} f(i) \leq \sqrt{\sum_{i=2}^{\frac{k}{R}} \mu(i)}$$

Proof 16 *The function $f(i)$ is derived from $\mu(i)$ by "enumerative combinatorics" approach, which list all the possible combinations of $f(i)$ for each $\mu(i)$. The combination is illustrated as follows:*

$\mu(2) = f(1)f(1) + 2f(2)f(0)$

$\mu(3) = 2f(2)f(1) + 2f(3)f(0)$

$\mu(4) = 2f(3)f(1) + f(2)f(2) + 2f(4)f(0)$

...

$\mu(i) = 2f(i-1)f(1) + ... + f(i/2)f(i/2) + 2f(i)f(0)$

Add up the left and right items of equations respectively, we can obtain:

$$\begin{aligned} \sum_{i=2}^{\frac{k}{R}} \mu(i) &\geq f(1)f(1) + 2f(1)f(2) + 2f(3)f(1) + ... \\ &= (f(1) + f(2) + ... + f(\frac{k}{R}))^2 \end{aligned}$$

The left side is larger than the right side because we omit the $f(0)$ item. Thus, we prove the

claim that:

$$\sum_{i=1}^{\frac{k}{R}} f(i) \leq \sqrt{\sum_{i=2}^{\frac{k}{R}} \mu(i)}$$

We now show that the expected value for the degree of encoding symbol after distribution deconvolution. By obtaining expected number of degree for encoding symbols, we further analyze the total message complexity compared with the case in which LT codes are applied individually in each node.

Theorem 17 *The recursive distribution deconvolution gives an upper bound on the expected number of encoding symbols in the network as* $O(\log N \cdot \frac{\sqrt{k}}{\ln(k/\delta)}) \cdot (k + c \cdot \sqrt{k} \cdot \ln^2(k/\delta))$.

Proof 18 *The essential idea is to prove the expected number of encoding symbols required for each node after degree distribution deconvolution does not exceed that before degree distribution deconvolution. We define an indicator variable* X_i *for each symbol, and* $X = \sum_i X_i$. *After j-th deconvolution, we assume the distribution function as* $f(i)$, *thus, the expected number of encoding symbol is computed as:*

$$\begin{aligned}
E_j(X) &= \sum_{i=2}^{\frac{k}{R}} f(i) \cdot i + p(1) \\
&\leq \sum_{i=2}^{\frac{k}{R}} f(i) \cdot \frac{k}{R} + p(1) && (4.5) \\
&\leq (\sqrt{\sum_{i=2}^{\frac{k}{R}} \mu(i)} + p(1)) \cdot \frac{k}{R} && (4.6) \\
&= (\sqrt{1 - \mu(1)} + p(1)) \cdot \frac{k}{R} && (4.7) \\
&< 2 \cdot \frac{k}{R}
\end{aligned}$$

where in (3), *we use the fact that* $i \leq \frac{k}{R}$, *the inequality holds by replacing each* i *in the expected number* $E_j(X)$ *with* $\frac{k}{R}$. *Inequality (4) holds true according to Lemma 1. Equality*

(5) *shows that the summation of probability distribution should equals to* 1. *As a result, the expected number* $E_j(X)$ *after j-th deconvolution is less than* $2 \cdot \frac{k}{R}$ *(*$R = c \cdot \sqrt{k} \cdot \ln(\frac{k}{\delta})$*). And in tree structure, the depth of network is* $O(\log N)$. *Since decoder of ONEC will observe the RSD, the number of encoding packets required in decoder side is* $k+c\cdot\sqrt{k}\cdot\ln^2(k/\delta)$. *Therefore, the total expected number of encoding symbols is* $O(\log N \cdot \frac{\sqrt{k}}{\ln(k/\delta)}) \cdot (k + c \cdot \sqrt{k} \cdot \ln^2(k/\delta))$, *where* N *is network size.*

[17] proves that the average degree of encoding symbol for LT codes is $O(\ln(\frac{k}{\delta}))$. When every node applies LT codes independently, its average message complexity is $O(\log^2 N \cdot \ln(\frac{k}{\delta})) \cdot (k + c \cdot \sqrt{k} \cdot \ln^2(k/\delta))$. The $O(\log^2 N)$ is due to the fact that every intermediate node forwards all packets from subset nodes. After distribution deconvolution, ONEC save considerable amount of overhead compared with LT codes.

Decoding Robustness. We use "And-Or Tree", an theoretic analysis tool proposed in [65], to give an analysis of the decoding robustness of ONEC. Specifically, it turns out to see what fraction of raw symbol data can be recovered when a portion of encoded packet are lost during communication. We consider every communication link of reliability φ, and the average hop-count in the network as h. Thus, the probability of lost encoded packets is expected to be $(1-\varphi)^h$. Since encoded data stream contain packets of degree 1, which can be directly recovered, and packets of higher degree ($d > 1$), which require the conducting of Belief Propagation algorithm on decoding graph. It is obvious from RSD analysis in [17] that the average number of packets of degree 1 equals to $c \cdot \sqrt{k} \cdot \ln(\frac{k}{\delta})$, and overall number of packets required for high-probability decoding is $c \cdot \sqrt{k} \cdot \ln^2(\frac{k}{\delta})$. Now, we denote the fraction of raw data which are not received as $\phi_L = \frac{(1-\varphi)^h}{\ln(\frac{k}{\delta})}$, and the fraction of missing encoded packet with degree ($d > 1$) as $\phi_R = (1-\varphi)^h \cdot (1 - \frac{1}{\ln(\frac{k}{\delta})})$.

Next, we gives some terminologies for decoding process. During belief propagation decoding, a bipartite graph is utilized to illustrate the procedure, with raw data symbols on the left and encoded packets on the right. Let $(p_0, p_1, ..., p_L)$ and $(q_0, q_1, ..., q_R)$ be the probability vectors, denoting that each node on the left has degree i with probability p_i, and each node on the right has degree j with probability q_j independently. According to the

"And-Or Tree" analysis, we first consider a subgraph G_s: choose an edge (u, v) uniformly at random from all the edges between left and right sides, then construct G_s as by taking the left node u and all its neighbors within certain distance $2l$ after deleting the edge (u, v) [65]. Then, for any left node i, $\alpha_i := \frac{ip_i}{\sum_{i=1}^{L} ip_i}$ is the probability that a uniformly chosen edge is adjacent to it; and $\beta_j := \frac{jq_j}{\sum_{j=1}^{R} jq_j}$ is the probability for a node on the right who is attached to the chosen edge. Define the polynomials: $\alpha(x) = \sum_{i=1}^{L} \alpha_i \cdot x^{i-1}$ and $\beta(x) = \sum_{j=1}^{R} \beta_j \cdot x^{j-1}$.

According to *Lemma* 1 in [65], in order to make the probability γ_l that the left node u can not be recovered approach 0 as l grows, the following condition needs to be true:

$$\varphi_L \cdot \alpha(1 - (1 - \varphi_R) \cdot \beta(1 - x)) < x(1 - \epsilon) \tag{4.8}$$

for all $x \in (0, \varphi_L]$, with a constant $\epsilon > 0$.

When we consider a dual inequality of the above problem, with the fraction φ_R of nodes on the right are lost, we then can get:

$$\beta(1 - \varphi_L \cdot \alpha(1 - (1 - \varphi_R)x)) > x(1 + \epsilon) \tag{4.9}$$

for all $x \in (0, x^*]$. Here, x^* is the largest value for which Inequality (9) holds. Therefore, we can draw the following theorem.

Theorem 19 *When there is φ fraction ($\varphi = \varphi_L + \varphi_R$, φ_L denotes lost fraction of degree 1 packets and φ_R for the lost fraction of packets of degree greater than 1) of encoded packets lost, the recovery ratio of decoding can be obtained by: $1 - \varphi_L \cdot P(1 - (1 - \varphi_R) \cdot x^*)$, where $P(x) = \sum_{i=1}^{L} p_i \cdot x^i$ and x^* is the maximum fraction of edges, to which right node can recover left node.*

Proof 20 *The value of x^* can be considered as the maximum fraction of edges between left and right nodes, so that each of right node v has all its neighbor nodes recovered except u, making u recoverable during decoding. And $\xi = (1 - (1 - \varphi_R) \cdot x^*)$ is the fraction of edges (u, v) between left and right which are not able to recover the value at node u. Therefore, at*

the end of decoding, a left node u with degree i has probability $\delta \cdot \xi^i$ *to be unrecovered. In total, there are* p_i *fraction of such node with degree i, and summing up all the nodes from degree* 1 *to degree L gives the results.*

4.1.3 Opportunistic In-Network Recoding

We recall that our goal is to achieve data collection of high reliability in challenged sensor network. The alive nodes status are intermittent and link connection is lossy. In such a disruptive communication environment, the data collection at the base station becomes partial, which might compromise or even halt the decoding procedure. ONEC design has observed this and as a result employ the opportunistic recoding to combat the disruptive network communications.

We then explain the idea for opportunistic recoding. Once every node derives its degree distribution, each source node produces encoded packet based on its own derived degree distribution. In forwarding nodes, a reception window (buffer) is set to receive packets for recoding. To note that, this buffer is only used to store the received packets inside window, and it is cleared as soon as the XOR is carried out and the encoded packet has been generated. In the good link communication, each intermediate node can receive more packets than those from all its children in the reception window, then generate recoded packets and forward them towards the sink. In the poor link connection, some packets from children can get lost, but some opportunistic overheard packets can fill the gap. According to the degree distribution deconvolution, the aggregated packet degree distribution eventually satisfies the RSD in the sink.

Illustrated in Figure 4.5, overheard link is denoted by dashed line, and tree routing path is in solid line. By on-path opportunistic recoding, if node 5 fails, node 3 can still overhear the packet from 7. If the node 5 comes alive, node 3 can selectively receive and recode the packet from node 7 by checking the symbol redundancy in the packet from node 5. However, if the node 1 fails, sink node will miss the the packets from node 3 and the nodes in its subtree even with on-path opportunistic XOR. It is because that only node 1 and 2 can

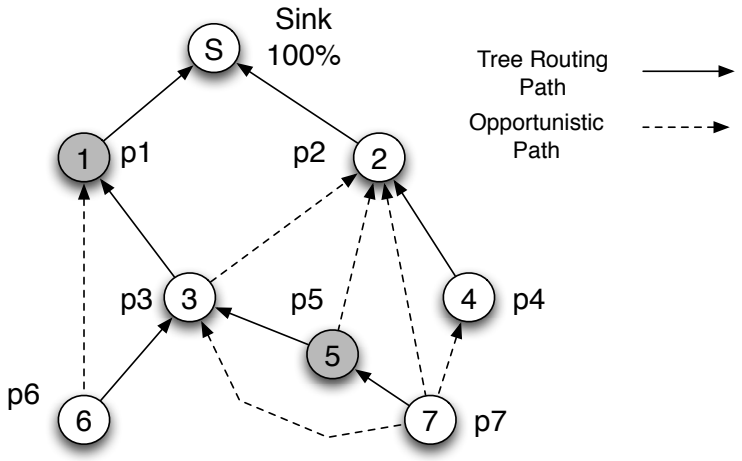


Figure 4.5 Opportunistic in-network recoding.

reach the sink in one hop. As illustrated, node 2 is able to overhear the encoded packet from 3 and 7 respectively. If we make full utilization of this overhearing over multiple paths to execute the opportunistic XOR in the forwarders, more innovative symbols can seep through the disruptive network to reach sink, and hence the data reliability can be largely improved.

The Algorithm 3 described the approach on how to calculate the probability that a node accept and recode the overheard packets.

Algorithm 3 Opportunistic XOR Algorithm

1: Receive an overheard encoded packet
2: Calculate the difference of hop-count: h_d
3: **if** Entire symbol is contained in current received packet **then**
4: Drop the symbol
5: **else**
6: The symbol is accepted with probability: $p(d_s)^{(1-1/h_d)}$
7: **end if**

In the Algorithm 3, d_s is the degree of innovative overheard packet and $p(d_s)$ is the probability for forwarder to select the symbol with degree of d_s, derived from its own degree distribution. And h_d is the hop-count difference between the source of overheard packet and current forwarder. The acceptance probability is proportional to hop-count difference. The reason is that the overheard packet would be more likely to reach the sink, by assigning more

probability to the nodes closer to the sink. And f_i equals $max\{(p(d_s))^{(1-1/h_d)}\}$ on the path i. In line 6 of Algorithm 3, if h_d equals to 1, the accept probability is 1, which conforms with the normal encoding scheme.

One practical issue of on-path opportunistic recoding scheme designs is that encoded symbol along different paths might get cancelled in converged points because of the simplistic XOR operation. The cancellation occur only when there are even number of nodes along the path who encode the same symbol into packet. In fact, in our simulation, the occurrence of symbol cancellation by XOR is rare.

Overall, with the opportunistic overhearing applied in the network, the original encoded symbol can have a considerably higher probability to reach sink.

4.1.4 Data Decoding

Finally, the code degree distribution of arrival packets in decoder is expected to be consort with the *Robust Soliton Distribution*. At the beginning of decoding, when an encoded packet arrive, it will be buffered for later decoding if it is not a packet with degree one. If a native packet (degree one) is received and recovered, it is placed in the ripple.

While the ripple size is above 0, a released symbol x in the head of ripple is processed, and every encoded packet y containing x is XOR-ed with x and hence the packet degree is reduced by 1. When the degree of an encoded packet decreases to 1, its value can be recovered. The value can be stored into ripple, as long as the symbol has not been processed and not in the ripple either. The decoding is successful if all the data is recovered.

4.2 Protocol Implementation Detail

We present the implementation of proposed ONEC protocol. The protocol resides in the routing layer, with a traditional converge-cast routing protocol sitting together. We adopt the Collection Tree Protocol (CTP) [66] to construct a tree structure for control message communication, including recursive degree distribution deconvolution, dynamic notification of symbol size fluctuation and incremental degree distribution update. Illustrated

in Figure 4.6, the ONEC protocol components (shaded parts) are implemented as flexible components, which co-work with TinyOS protocol stack. When enabled, it takes charge of opportunistic routing, recoding and forwarding to improve the data collection performance.

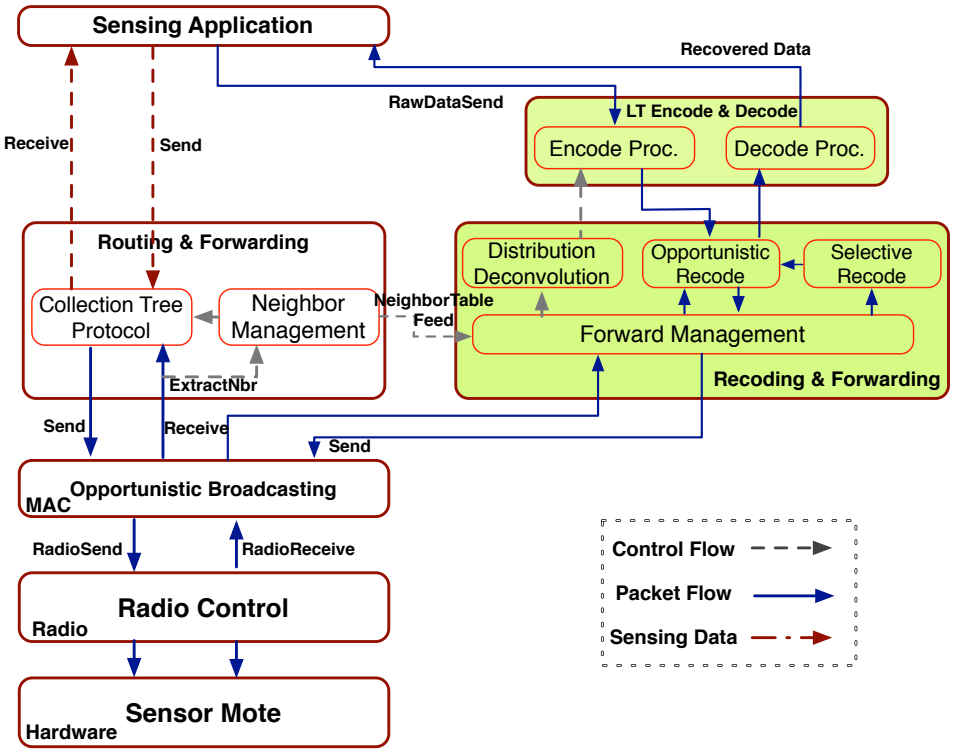


Figure 4.6 Software Implementation on TinyOS, with shaded part as our ONEC components.

Besides, we employ *M&M* Markov Model [67] to simulate the wireless link in radio stack. TOSSIM comes with different models simulating several radio characteristics, in which CPM model [68] models the variations in radio signal strength using a statistical noisy reading traces. However, *M&M* Markov Model [67] utilizes Hidden Markov Models (HMMs) and Mixtures of Multivariate Bernoullis (MMB) to mimic the long and short time scale behavior of lossy wireless links. Multi-level Markov Models characterize the data traces based on packet reception rate (PRR), which provides much better matching between simulated model and the time-varying data traces collected from real testbed. We settle the system parameter $Q_1 = 6$ in *level-1 hidden Markov model* (L1-HMM), which represent Q_1 transitions between long-term states; $Q_2 = 2$ in *level-2 hidden Markov model* (L2-HMM); and mixture component

$M = 20$. In terms of LT coding, we adopt the system parameters $\delta = 0.05$, which allows for decoding success probability $(1 - \delta) = 95\%$ with sufficient packets received, and $C = 0.1$ respectively.

4.3 Performance Evaluation

In this section, we have evaluated the performance of our proposed $ONEC$ based on simulation experiments. The experiments are conducted on TOSSIM [69], the TinyOS [70] simulator. Experiments are conducted in large-scale networks by varying network sizes, input symbol sizes, link loss rates and node failure percentages.

4.3.1 Simulation Setup

We consider a random graph of network size N, where nodes construct a data collection tree for distributed degree deconvolution. We deploy the network in a random fashion, so that each node in the constructed coding tree contains different subtree sizes. Each node i is able to generate certain amount of data and is required to deliver them to the base station reliably. The packet generation rates in each node (denoted as R_i for node i), are heterogeneous across the network. The generated packet is comprised of different number of raw *innovative* neighbor symbols, which are encoded by XOR operation. We also define the tree level of node i as the shortest hop count from node i to sink, so that we can inspect the transmitted packet number level by level along the data collection tree. Since each node has different sub-tree sizes, without loss of generality, we unify the input symbol size k of each node. The size k varies from 50 to 500. The performance is evaluated mainly by the metric of message complexity. Here, the message complexity is the expected transmitted number of packets (including the control messages), that is needed for reliably decoding data from network. The result is the average of the 500 runs of the simulation experiment.

Performance of $ONEC$ is compared with other coding schemes, under the same network conditions: TCP: it only uses TCP protocol to realize reliable data delivery. LT: it simply applies the LT encoding in every source node, but no network coding in the intermediate

nodes. *MORE* [4]: Random linear encoding is used in the forwarding nodes, and Gaussian Reduction is applied in decoder. *GROWTH* [35]: the *GROWTH CODES* increases the degree of encoded packet is growing over time, which make sure that increase is in efficiency as data accumulates at the sink. *CCACK* [5]: a node can suppress the redundant packets of its neighbors, by broadcasting the acknowledge message. The duplicated dependent encoded packets are reduced by CCACK. *SlideOR* [34]: SlideOR applies random linear codes and utilizes the sliding window to encode intra-flow data traffic. The sliding window advances only in the source upon receiving the ACK feedback from decoder. The ACK message is assumed to be reliable in the design.

4.3.2 Network codes validation

Before we evaluate the performance of decomposed distribution in the network, we first validate the correctness of $ONEC$ codes evaluating on the received degree distribution for decoding. We validate the distribution of total received degree. Figure 4.7 shows the comparison, in which the top figure indicates the packet degree distribution received from single node running RSD, and the bottom figure shows the actual received degree distribution from a network running the $ONEC$. The symbol size k is preset as 50 for the entire network and single node as well. It can be observed that the received degree distribution matches the Robust Soliton Distribution. Moreover, we validate the total number of received packets required to decode the raw data set with the probability at least $(1 - \delta)$. The simulation results are shown in Figure 4.8. $ONEC$ presents constantly close performance to RSD when varying the input symbol size k. Thus, the correctness of decomposed degree distribution is validated.

4.3.3 Communication evaluation

We conduct simulations to evaluate the impact of various factors on the data decoding probability, including total message cost, network size and symbol size. The comparison results show the advantages of $ONEC$ coding, which has consistent performance with the

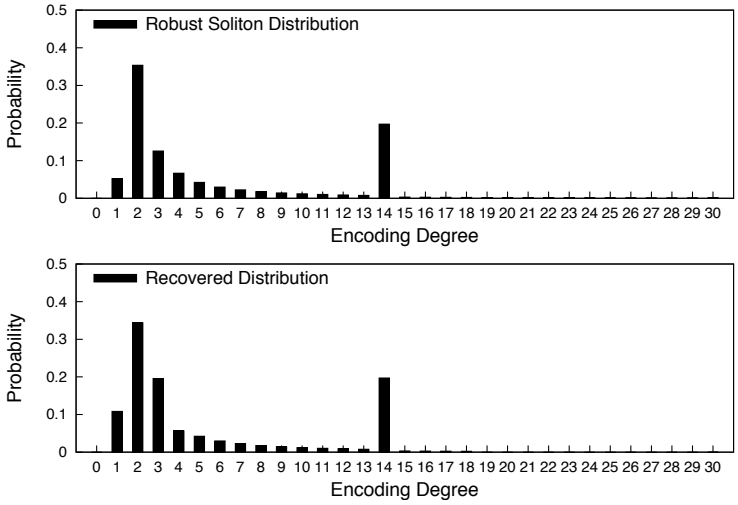


Figure 4.7 Opportunistic network erasure codes validation.

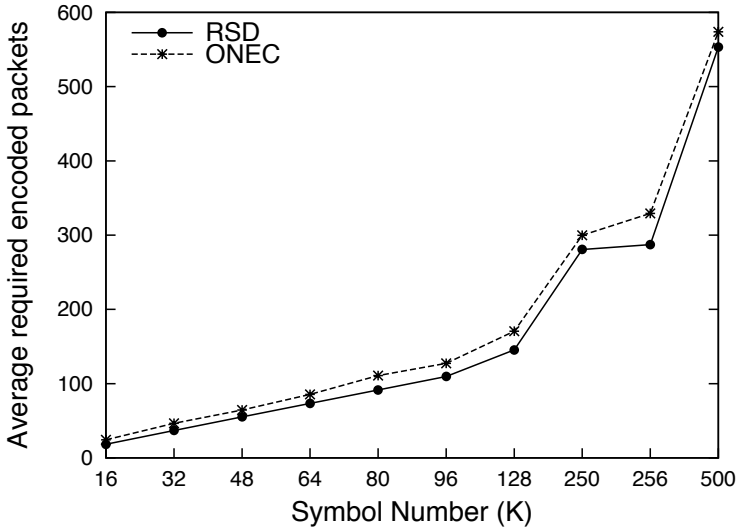


Figure 4.8 The number of encoded packets needed to decode the raw data set is compared between RSD and ONEC.

scaling factors.

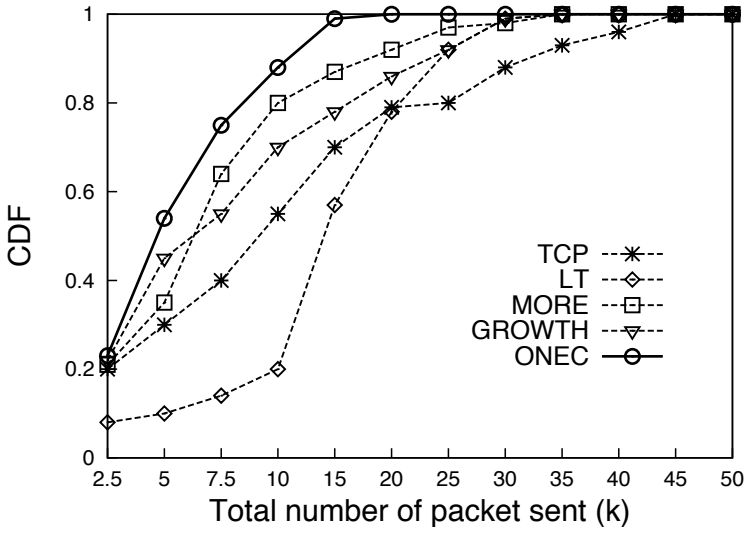


Figure 4.9 CDF of decoding success probability under different sent packet numbers.

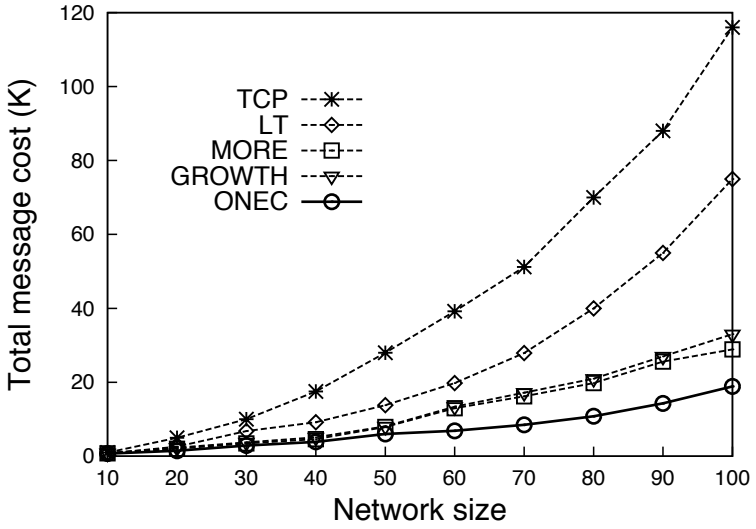


Figure 4.10 Total amount of packet transmissions required under different network size.

Impact of message cost We evaluate the impact of message cost on decoding by varying the number of sent packets under the network size of 100, with 50 symbols in each node. The link reliability is assumed as 20%. The simulation is conducted in random tree topology with 100-time repetition. The results illustrated is measured in average value. In Figure 4.9, the

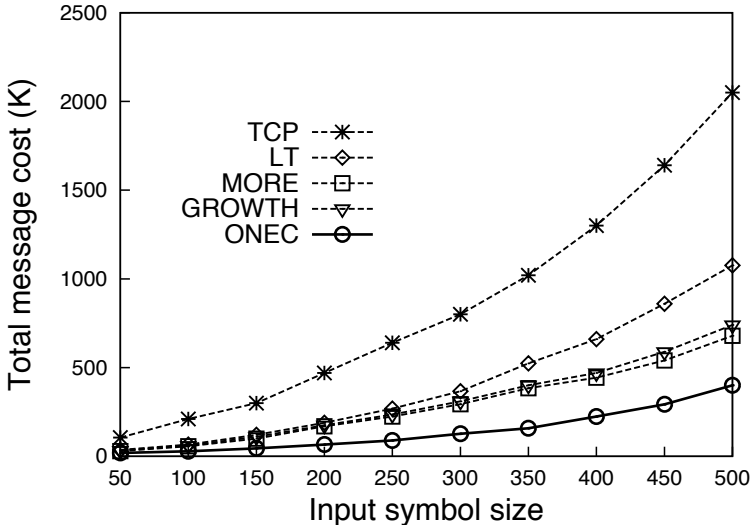


Figure 4.11 Total amount of packet transmissions required with varying sizes of input symbol.

number of total sent packets during evaluation ranges from $2,500$ to $50,000$. It unveils that $ONEC$ has higher decoding probability than other coding schemes, at any given number of packet sent. $ONEC$ achieves 100% decoding with message cost of $20,000$ packets, which is at least one-third less than $MORE$, $GROWTH$ and LT codes. Though the decoder expect the same degree distribution from $ONEC$ and LT codes, the decomposed degree distribution and opportunistic recoding help $ONEC$ reduce considerable amount of redundant message cost in the entire network. Thus, from the network perspective, $ONEC$ considers the network as an integral source, which saves message from redundant encoded messages and shift the "increasing leap" of LT codes earlier.

Since shifted earlier, the performance of $ONEC$ also surpass the $GROWTH$. For $GROWTH$, it makes the optimal degree changing point to accommodate the decoding pace. But, the considerable amount of neighbor data exchanging make the message cost high.

Comparing to $MORE$, $ONEC$ still has performance advance as the opportunistic recoding take place with different probability based on hop-count discrepancy. But, $MORE$ makes nodes in the forwarding list encode the packets and forward them, which has slightly more message cost. We evaluate their performances in various network environment, including disruptive networks in subsequent sections.

Another observation from Figure 4.9 is that TCP has similar decoding probability as $ONEC$ codes when the total $2,500$ packets are sent at the beginning. With the increasing number of packets sent, decoding probability of TCP grows slowly, ending in $45,000$ packets for 100% decoding. It is because TCP uses end-to-end acknowledgement mechanism to guarantee delivery, which cause a lot of message waste in the ACK messages.

Impacts of network size

We have evaluated the performance of TCP (no coding), LT codes, $MORE$ and $ONEC$ in the randomly deployed network, with $k = 50$ symbols in each node. In Figure 4.10, we measure the average number of transmitted packets (which is required for successfully decoding raw data) in the networks of varying sizes. It can be observed that $ONEC$ outperforms all other coding schemes. For TCP, which needs end-to-end acknowledgement message to enable reliable transmission, the message complexity is expected to grow exponentially with the network size. TCP has a message complexity of 50% more costly than that of the LT codes, and twice as many as that of $ONEC$. LT codes apply erasure coding only on source nodes, such that forwarded packet number is still exponentially related to the network size. With increasing network size, the packet number required for TCP and LT codes increases much faster than $ONEC$. This indicates that we obtain more benefits with the $ONEC$ schemes when the network size increases.

Applied to a large-scale network, $ONEC$ gains its increasing benefit margin, since the $ONEC$ enables opportunistic recoding in intermediate nodes. Its transmitted packet number has a linear relation with the network size, which results in a smaller growing rate. $MORE$ utilizes random linear coding and opportunistic recoding in forwarding nodes. It has smaller message complexity than LT and TCP schemes. However, $MORE$ still asks for more transmitted packets than $ONEC$ in different network size to decode the raw data reliably. As in $MORE$, every node belonging to the forwarding list of a specific packet needs to encode it, as long as the packet is "innovative". $ONEC$ distinguishes the overheard packet by difference in hop-count, and recodes them with different probability. Also it can reduce the redundancy of transmitting similar encoded symbols. The packet cost for $GROWTH$

codes also maintain higher than $ONEC$ under different network size. The reasons are two-folded: 1) in $GROWTH$, the exchanging message is considered as a waste if the "innovative" data in the neighbor already in the codeword; 2) the decent decoding performance relies on a good "mixing" of random selected data, which requires a considerable amount of message exchanging.

Impacts of input symbol size Next, we examine the impact of various input symbol sizes in the same network size of 100. Since randomly deployed network has heterogeneous subtree size for each node, without loss of generality, we assign the same symbol size to each node. In Figure 4.11, it shows the similar trend in performance margins of $ONEC$ as the one illustrated in Figure 4.10. Moreover, $ONEC$ has more advantages when input symbol size increases.

4.3.4 Energy and resource evaluation

Besides decoding performance, energy efficiency is also a major evaluation metric for network coding scheme to improve data reliability in a disruptive network. The energy consumption contains three parts: (1) communication cost of data packets; (2) cost of control packets; (3) computation cost for encoding and decoding. Both radio communication costs are evaluated by counting the amount of packets transmitted and received inside network. Energy consumption is calculated by considering the transceiver model of CC2420, where current consumption for radio transmitting is $17.4mA$, and $19.7mA$ for radio receiving. Computation cost is calculated based on TI MSP430 MCU. We evaluate these metrics in different network coding schemes of TCP, $MORE$, $GROWTH$, $SlideOR$, $CCACK$ and $ONEC$ when 1000 symbols are successfully decoded, and normalize energy consumption to TCP.

First, TCP uses no coding scheme, and its ACK control message takes more than 50% in the final energy consumption. The reason that TCP consumes more energy than any other coding schemes is that the lossy ACK transmission deteriorate the efficiency of sliding window. TCP's computation cost can be neglected compared with coding methods.

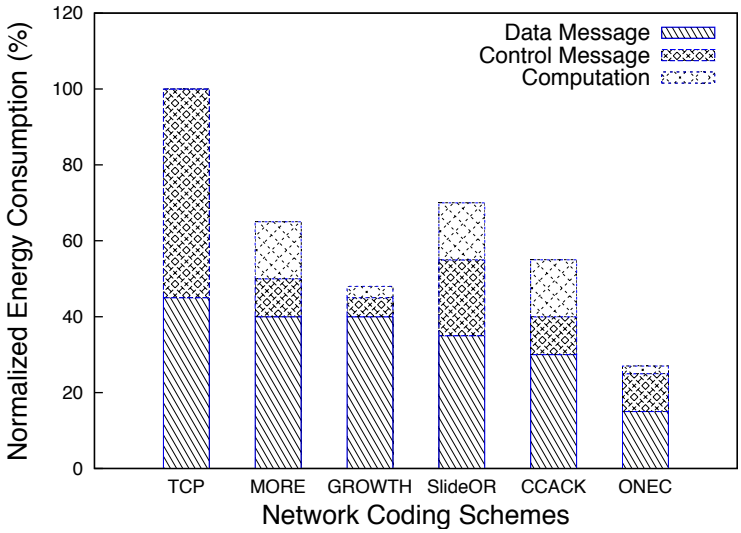


Figure 4.12 Energy consumption of different coding schemes

Second, the approach using random linear coding, e.g. $MORE$, $SlideOR$ and $CCACK$, requires more sophisticate encoding and decoding, which entails more energy consumption for computation. In contrast, the $GRWOTH$ and $ONEC$ are based on erasure codes, which demands much less computation complexity. Third, in terms of control message, $ONEC$ is not the best, since it needs to recursively decompose the degree distribution along forwarding paths, and basic beacons are needed to maintain a neighbor list. The $GROWTH$ codes consume less energy consumption, since it is designed for zero-configuration network with lossy link condition. Finally, $ONEC$ has the least energy consumption for data transmission. In other words, $ONEC$ introduce less data redundancy for decoders. We can see that $CCACK$ has less energy cost for data transmission than $MORE$, since it introduces NSB, which suppresses redundant data. $SlideOR$ can also introduce data redundancy if the window advances inappropriately due to missing ACK.

Figure 4.13 evaluates the recoding buffer size in the forwarding nodes. In Figure 4.13, it is shown that $ONEC$ has much less buffer size requirement for intermediate nodes. However, $GROWTH$ and $MORE$ consume a considerable amount of buffer, with about 270 bytes and 220 bytes under network size 100 respectively. The buffer is required for $GROWTH$ to store the received packet in the past to reconstruct a newly encoded packet. $MORE$ stores received

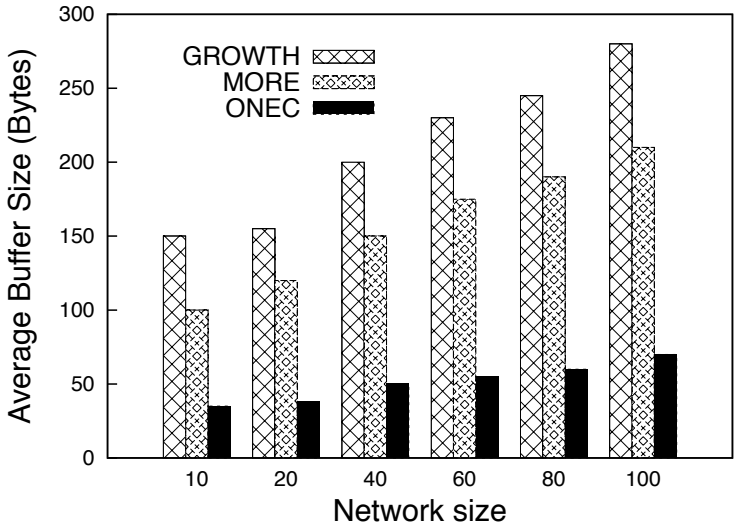


Figure 4.13 Average buffer size in forwarding nodes

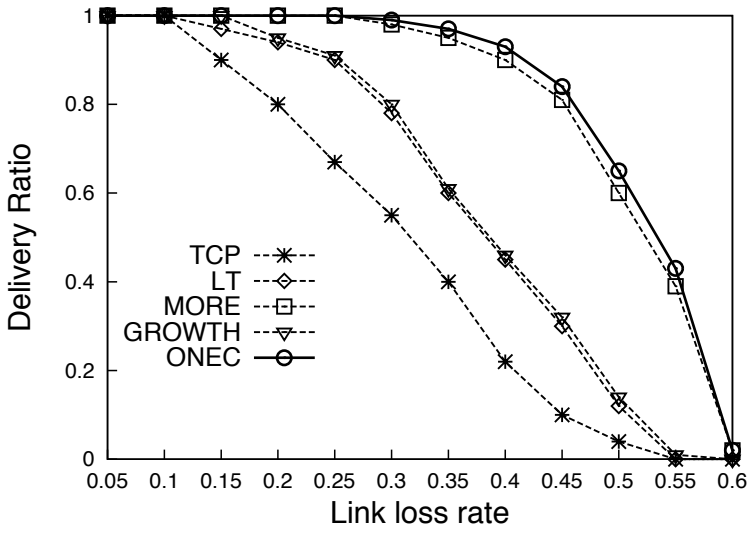


Figure 4.14 Packet delivery ratio under different link loss rates.

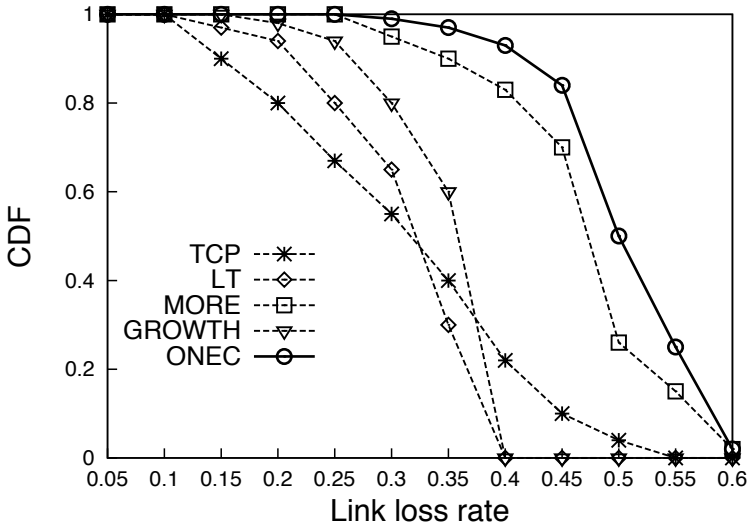


Figure 4.15 CDF of decoding success probability under different link loss rates.

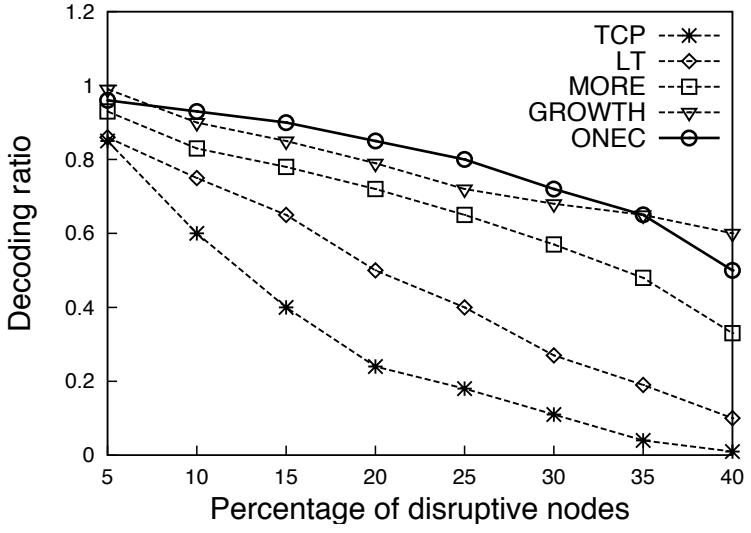


Figure 4.16 Symbol decoding ratio in the disruptive networks.

data for checking if the packet is "innovative". However, $ONEC$ is an opportunistic network coding scheme that has very little demands on the buffer size, since it does not need to store the history packet either for "innovative" packet check or for generating newly encoded packet.

4.3.5 Robustness evaluation

Now we study robustness of $ONEC$ and other coding schemes in the disruptive network environment, which includes node failures and lossy link communications.

Impacts of link loss rate $ONEC$ is designed and developed for improving data reliability in disruptive sensor networks. Now we evaluate its feasibility and performance in different networks with diverse link reliabilities. The network size is 100 with symbol size 50 on each node. Figure 4.14 and Figure 4.15 indicates the probability of delivery ratio and decoding success under different link loss rates respectively. Sink node starts to decode or check the completeness of received data, when total amount of transmitted packet in the network reaches $100,000$.

Figure 4.14 illustrates the delivery ratio of different coding schemes under various link loss rates. $ONEC$ and $MORE$ have the similar performance, while the LT and $GROWTH$ have worse delivery ratio. The reason is that the opportunistic forwarding or recoding help more symbols seep through the network to reach sink. However, larger delivery ratio can not guarantee the higher decoding probability, as it contains much redundancy in the packets seeping through network.

In Figure 4.15, it is not hard to discover that both LT and TCP have the earliest drop at the link loss rate of 15%. However, $ONEC$ can maintain more than 90% decoding success ratio until the link loss rate degrades to 40%. $MORE$'s performance is between LT and $ONEC$. The reason why $ONEC$ can outperform the $MORE$ scheme is that $ONEC$ is free of maintaining the forwarding list for each packet. In the disruptive network, nodes' status are transient and disruptive. Therefore it is difficult to keep the forwarding list updated every time packet is about to be transmitted. Some recipient node will drop the packet

simply because it does not belong to the forwarding list of the received packet, wasting the chance to recode the packet. $ONEC$ utilizes the symbol degree probability and hop-count between transmitting node and recipient node, and determine the probability of "recoding" received symbols into current packet.

For $GROWTH$ codes, the decoding probability drop dramatically when the link reliability is under 70%, and there is almost no successful decoding when the link loss rate reach 40%. Since $GROWTH$ codes hard-coded the changing point $k_1, ..., k_i, ..., k_n$ to the nodes before deployment, the lost packet could significantly hamper the decoding process. As insufficient packet number of degree k_i will stop the decoding of k_{i+1}, the $GROWTH$ fail to decode the rest of the packets at the point of 40% loss rate. With the link loss rate increases, the TCP gets much worse in an exponential fashion due to the ACK implosion effect. Thus, it is not the appropriate solution to guarantee data reliability in the disruptive networks.

In Table 4.1, we further evaluate the decoding performance of $ONEC$ in lossy communication network by comparing with other recent works, including $CCACK$ and $SlideOR$. The network size is 100, with varying symbol size on each node from 50 to 200 bytes. We set the check point for decoding performance evaluation when network message number υ achieves $100,000$, $210,000$ and $470,000$ for symbol size 50, 100 and 200 respectively. Lossy link is simulated by 2-Level Hidden Marcov Model, with varying loss rate from 5% to 60%, which prohibit most of decoding procedure from success.

The first thing observed from Table 4.1 is that $ONEC$ has a better overall decoding performance than $CCACK$ and $SlideOR$ in varying loss rate conditions. Particularly $SlideOR$ has the worst behavior and fail in 100% decoding when link reliability drops to the rate 90%. It is due to the fact that $SlideOR$ relies on a reliable ACK feedback to advance the encoding window. In other words, the loss of ACK message will halt the encoding window and degrade decoding performance as well. Nodes of $CCACK$ can suppress the transmission of redundant encoded packets, which gives it a better throughput over $MORE$. However, it

Table 4.1 Decoding comparisons with different symbol sizes under lossy links

Link Loss Rate	LLR Mean±StdDev	Proposed ONEC Symbol Size			CCACK Symbol Size			SlideOR Symbol Size		
		50	100	200	50	100	200	50	100	200
0.10	0.101±0.012	1	1	1	1	1	1	0.98	1	1
0.20	0.202±0.029	1	1	1	1	1	1	0.90	0.91	0.91
0.30	0.301±0.012	0.99	0.99	1	0.95	0.96	0.97	0.65	0.67	0.68
0.40	0.405±0.011	0.93	0.95	0.96	0.83	0.85	0.84	0.45	0.45	0.46
0.50	0.50±0.018	0.50	0.56	0.57	0.26	0.31	0.33	0.13	0.14	0.14
0.60	0.60±0.020	0.02	0.12	0.20	0.03	0.05	0.07	0.01	0.01	0.02

is subject to the forwarding list maintenance cost of $MORE$. The advantage of $ONEC$ is that it is free of updating the forwarding list, received packets are encoded into packet with a probability respective to the hop-count. Second, the advantage of increasing symbol size is not significant, and performance gain decreases when link reliability drops.

Impacts of disruptive nodes

In the previous discussion, we show the advantage of utilizing opportunistic recoding in disruptive networks. $ONEC$ enables to recode any overheard symbol packet with a probability. Thus, forwarding list is not necessary for each transmitted packet. Figure 4.16 shows symbol decoding ratio of three schemes under different percentage of disruptive nodes. In Figure 4.16, the total number of sent packets in each node is fixed. The figure illustrates that a decoding ratio of average 85% is reached by $ONEC$ in the network of 15% malfunctioned nodes. In other words, all the symbols except the ones from failure nodes are successfully decoded in sink, by enabling opportunistic recoding. With the percentage of disruptive nodes increasing, the decoding ratio of $MORE$ decreases faster than the $ONEC$. The reason is that more disruptive nodes in the forwarding list, $MORE$ will become less efficient. That is to say, the forwarding list of $MORE$ is not fully capable of capturing the node status in presence of disruptive communication.

In Figure 4.16, we observe that $GROWTH$ codes can render better decoding probability than any other schemes, including $ONEC$, in the case of large percentage of disruptive nodes. It is because the exchanging messages used in $GROWTH$ are reduced when the neighbor size decreased. Moreover, the message cost of achieving good "mixing" is decremented as well.

CHAPTER 5

DATA COMMUNICATION WITH ENERGY-SYCHRONIZED NETWORK CODING (ONCODE)

Renewable energy holds great promise of making wireless sensor network truly battery-less and have perpetual lifetime. However, the amount of harvested energy has dynamic and volatile patterns, since it depends on many uncertain environmental factors (such as weather, light intensity, temperature etc.). *Energy synchronized design* is the key to meet the lifetime requirement in battery-less sensor networks, preventing the residual energy from being overdrawn. The term *energy synchronized design* means the node operations (including radio communication, sensing and computing) need to synchronize with energy fluctuation.

In the literature, several works (such as in [2]) adapt radio duty cycle in MAC protocol to different energy constraints. Other existing methodologies, such as opportunistic routing and joint energy-routing optimization were proposed to deliver data collaboratively. Opportunistic routing is adopted [3–5] to take advantage of multiple transmission opportunities to improve data throughput. Data source nodes demand explicit ACK feedback message from receiver to advance to next data blocks. Relying on ACK message for reliable data delivery becomes very difficult (if not impossible) in a disruptive multi-hop network. More importantly, those solutions often underestimate the effects of dynamic energy fluctuations in the nodes. Joint optimization [6] considers energy and routing together for the decisions and gives an optimal solution. But joint optimization requires pre-determined communication pattern and energy information and intensive computation cost, which is often not practical in real applications since energy fluctuation is not easy to predict. In an energy-harvesting sensor network, a good data delivery protocol shall immunize network from the unpredictable disruptions caused by energy fluctuations, while utilize opportunistic communications for reliable data delivery.

In this paper, we present the design, implementation and evaluation of *OnCode* protocol. *OnCode* is an opportunistic in-network coding and data delivery protocol. OnCode adapts the network coding and probabilistic forwarding *in-situ* to energy variations. It exploits probabilistic coding gain without relying on any predefined network structure and achieves high data delivery ratio without need of end-to-end feedback. It also self-tunes the encoding block to adapt to dynamic source data rates and ON/OFF wake-on ratios [1] from ambient neighbor nodes. Both theoretical analysis and experimental evaluations are conducted to demonstrate the performance of *OnCode*. To the best of our knowledge, *OnCode* is the first work that explores opportunistic network coding to synchronizes data delivery operation with dynamic nature of renewable energy sources, and does not depend on any particular routing structure underneath the network.

The rest of the paper is organized as follows. In section 5.1, we summarize the preliminary backgrounds. Then we present the design and analysis of *OnCode* protocol in section 7.1. In section 5.3, we discuss the details of hardware testbed and protocol implementation. In the following section 7.2, we present evaluation results in both testbed and simulation. We describe the related works in section 5.5, and conclude our work in section **??**.

5.1 Preliminaries

In this preliminary section, we present the network model in our problem and background knowledge about erasure codes and distribution deconvolution. Table 5.1 shows the notations used in the subsequent sections.

5.1.1 Network Model

We study the problem in a random network of node size n, where each node is powered by a renewable energy source, which has unstable energy harvesting rate. Because nodes' operations need to be synchronized with energy fluctuation, which causes nodes' activities

[1] Wake-on ratios denotes the portion of time radio is kept on.

Table 5.1 List of notations in protocol design and analysis

$\mu(*)$	Raptor Codes degree distribution
$\omega(*)$	Deconvoluted degree distribution
n	Size of network
m	Size of data set
D	Average node degree
w_i	Disruption ratio of node i
H_{ij}	Hitting time from node i to j
L_{ij}	Hitting latency from node i to j
R_i	Estimated number of delivered packet for node i
B_i	Encoding block size in node i
S	Total amount of innovative symbols in the network
d_i	Degree drawn from distribution $\omega(*)$ in node i
d_p	Packet degree drawn from distribution $\mu(*)$ for packet p
d_r	Packet degree for parity check degree
λ	Redundancy ratio of transmitted packets
ρ	Packet forwarding probability
Λ	Node priority

to become intermittent. Intermittent behaviors cause nodes to be disruptive from networks, which is denoted as disruption ratio w_i in node i.

In such a disruptive network, the opportunistic routing [3] is utilized to take advantage of the nature of radio broadcast in wireless communication environment to improve the data delivery efficiency. It has been shown that even if all data losses are correlated in different paths, the packet delivery probability still increases by some "lucky" transmissions, which reach further nodes than others. In opportunistic forwarding, multiple recipient nodes may overhear the same packet, where the node of highest priority forward the packet. The packets rebroadcast by forwarder suppress the rebroadcasting of the same packet from other nodes in the forwarding list. And the transition matrix of packet during opportunistic routing in each node can be defined as follows:

$$P_{ij} = \begin{cases} q_{ij} \cdot \Pi_{\Lambda_k > \Lambda_j}(1 - q_{ik}), & j \in N(i) \\ 1 - \sum_{j \neq i} P_{ij}, & i = j \\ 0, & j \notin N(i) \end{cases} \tag{5.1}$$

where $N(i)$ denotes the set of neighbors of node i, Λ_k is the priority of node k, and q_{ij} is packet transmission reliability from node i to node j.

5.1.2 Erasure Codes

Erasure codes enable the recovery of all the m symbols by decoding a sufficient amount of $m(1+\epsilon)$ encoded packets with a high probability. The XOR based encoding and decoding methods are effective and computation efficient. Unlike other random linear codes, the computational cost of erasure codes is more suitable for networks of low computational power devices. It has also been employed in many challenging applications, especially those environments with narrow feedback channels, since erasure codes improve the data throughput by removing dependence on ACK feedback control messages.

The key to the erasure codes is the degree distribution, which describe a probability distribution of encoding degrees among encoded packets. However, degree distribution is for end to end transmission, because encoding and decoding only happen in end systems. In order to apply erasure codes to network coding, we need to decompose degree distribution from end system to network nodes. The technique we use is deconvolution. The principle of using deconvolution in degree distribution decomposition is studied in previous works [36]. Generally, given a degree distribution $\mu(*)$ as an output function, node u takes two independent input encoded data streams from node v and w respectively, and tries to produce an encoded packet stream with degree distribution conforming with output function. Since node u XORs the incoming encoded symbols, the output degree in each packet is the sum of the degrees of those incoming stream. Moreover, node v and w generate their encoded symbols in an independent way based on their own degree distributions. Thus, the "recoded" packet degree conforms with the convolved function of distribution $\mu_u[i] = (\omega_v \cdot \omega_w)[i] = \sum_{j=1}^{i} \omega_v[j] \cdot \omega_w[i-j]$, where $\omega_v(*)$ and $\omega_w(*)$ are two input degree distributions.

Algorithm 4 OnCode Data Delivery

Input: Innovative sample data set $\mathcal{M}= \bigcup_{i=1}^{N} \mathcal{M}_i$, disruption ratio $w_{i,t}$ and degree distribution $\mu(*)$
Output: Encoded packets stream received by sink node: $\mathcal{P} = \bigcup_i \{\hat{p}_1, \hat{p}_2, ..., \hat{p}_t\}$ ($\forall\ i \in V$ & $\forall\ t \in [1, \tau]$)

1: $\mathcal{P} = NULL$;
2: **for** $t = t_1 \rightarrow t_T$ **do**
3: **for** $i \in V$ **do**
4: $\bigcup_i \{p_1, ..., p_t\}$= ASE (S_i, $w_{i,t}$, $\mu(*)$); (Adaptive-Source-Encoding)
5: **end for**
6: $\mathcal{P}^{'} = \bigcup_i \{p_1, ..., p_t\}$
7: $\mathcal{P}^{'} = OINE(\mathcal{P}^{'}, \bigcup_{i=1}^{N} \mathcal{M}_i)$; (Opportunistic-In-Network-Recoding)
8: $t \leftarrow t + \Delta T$;
9: **end for**
10: $\mathcal{P} = \mathcal{P}^{'}$

5.2 Algorithm Design and Analysis

OnCode is an energy-synchronized network protocol, designed for data delivery over unpredictable network disruptions. OnCode adapts the delivery and coding strategy *in-situ* to energy distribution, maximizing quality of data delivery in disruptive sensor network.

5.2.1 Design Overview

Algorithm 4 illustrates OnCode data delivery. *First*, source node exploits Adaptive Source Encoding (ASE), using a set of sample data $\mathcal{S}_i$, disruption ratio $w_{i,t}$ and degree distribution $\mu(*)$ as input. In $G = (V, E)$, each node $i \in V$ generates encoded packets and forwards them on probabilistic links, as shown in line $3 - 5$. *Second*, during data delivery cycle T, Opportunistic In-Network Encoding (OINE) is applied whenever an opportunistic packet is overheard as shown in line $6 - 8$. Payloads of encoded packets are recoded with the random portions of local data set in nodes decided by local degree distribution $\omega(*)$. This procedure repeats until packet degree d_p is satisfied. Output of OnCode is a sequence of encoded packets, which can be decoded with high probability. OnCode avoids the reliance on a specific routing structure and makes coding procedure of low complexity.

Algorithm 5 Adaptive Source Encode: Algorithm running on a source node i ($i \in V$)

1: **Constant**: $\mu(*)$, λ, ρ, ϵ, Δ_i
2: **Variable**: R_i, $\omega(*)$, M, M_i, $w_{i,t}$, $\mathcal{B}_i$
3: **for all** $j \in V$ **do**
4: Construct matrix P_{ij}; /*See Equation (5.2)*/
5: **end for**
6: Compute hitting latency $L_{i,sink}$; /*See Equation (5.3)*/
7: Derive packet reachability R_i;
8: Decide encoding block size: $B_i = \frac{R_i}{(1+\epsilon)\cdot\lambda}$;
9: Compute $\omega(*) = f(\mu(*), B_i, M_i)$; /*See Equation (5.4)*/
10: **for** $t = t_1 \rightarrow t_T$ **do**
11: $\mathcal{P}'_i = \mathcal{P}'_i \bigcup EFU(\mathcal{B}_i, \omega(*))$; /*EFU: Encode &Forward Utility*/
12: $t \leftarrow t + \Delta T$;
13: **end for**
14: Move to next encoding block.
15: UPON receiving energy update, go to Step 1.
16: Go to Step 8.

5.2.2 Adaptive Source Encoding

The procedure of adaptive source encoding in each node is illustrated in Algorithm 5. *First*, packet forwarding reliability is drawn to establish the state transition matrix in line $3-5$. *Second*, packet reachability to sink is estimated in line $6-7$. *Third*, an appropriate encoding block can be derived and the local degree distribution $\omega_i(*)$ is deconvoluted for source encoding (line $8-9$). Line $10-13$ encodes and forwards the data using EFU(*). EFU is a custom routine designed to generate encoded packet based on selected block and forward them over probabilistic links.

Packet Forward Probability We denote the probability of successfully transmitting a packet from node i to node j as q_{ij}. The parameter q_{ij} is only determined by the node disruption ratio in nodes i and j respectively. Let w_i denote disruption ratio for node i. Node i obtains the w_j from its neighbor nodes during initialization phase. Then node i calculates $q_{i,j}$ by the following equation: $q_{ij} = (1 - w_i) \cdot (1 - w_j)$.

Each node has its priority determined by individual node disruption ratio. For instance,

node i has higher priority than node j, if $(1-w_i) > (1-w_j)$. Node i knows its priority ranking among its neighbors by sorting the w_j among neighbors. Then nodes elect themselves to forward the packet from node i with the probability as follows:

$$P_{ij} = \begin{cases} q_{ij} \cdot (1 - \rho \cdot \log(\Lambda + 1)), & j \in N(i) \\ 1 - \sum_{j \neq i} P_{ij}, & i = j \\ 0, & j \notin N(i) \end{cases} \tag{5.2}$$

where ρ is the forwarding probability, Λ is the priority ranking of neighbor node j among its own neighbors. Node j is elected as forwarder with a probability of $q_{ij} \cdot (1 - \rho \cdot \log(\Lambda + 1))$. If Λ equals to 0 (denoting node j has the highest priority) node j forwards packet as long as it received it from node i with probability of q_{ij}.

Packet Reachability Estimate Every node estimates its packet reachability to base station based on updated wake-on ratios of neighbor nodes. For estimation purpose, the duration ratio of radio ON and OFF mode of neighbor nodes is sufficient, no further demand for the exact time slots. The estimation of packet reachability in node i is to calculate hitting latency L_{ij}, based on transition probability matrix P_{ij}. L_{ij} is the expected transmission delay between node i to j.

Figure 5.1 illustrates an example of radio wake-on in a case of sender i and receiver k. Time difference T_{ij} between T_{arrive} when a packet arrives at sender i and $T_{forward}(k)$ is denoted as the one-hop transmission delay. We do not assume specific wake-on schedules applied. w_i is the disruption ratio of node i in a cycle of time stretch τ. The wake-on point is uniformly distributed during time τ. $T_{ik}^{'}$ is the delay decided by wake-on interval in node i:

$$T_{ik}^{'} = \sum_{X=0}^{\lfloor (w_i) \cdot \tau \rfloor} X \cdot (\tau - X) \cdot (w_i)^X \cdot (1 - w_i)^{\tau - X}$$

where delay time X ranges from 0 to $\lfloor (1 - w_i) \cdot \tau \rfloor$, which is the maximum dormant time stretch of node i. Inside the cycle T, there are $(\tau - X)$ time segments with dormant length of X. $T_{ik}^{''}$ can be derived from similar equations, replacing w_i with w_k as the disruption ratio

in node k.

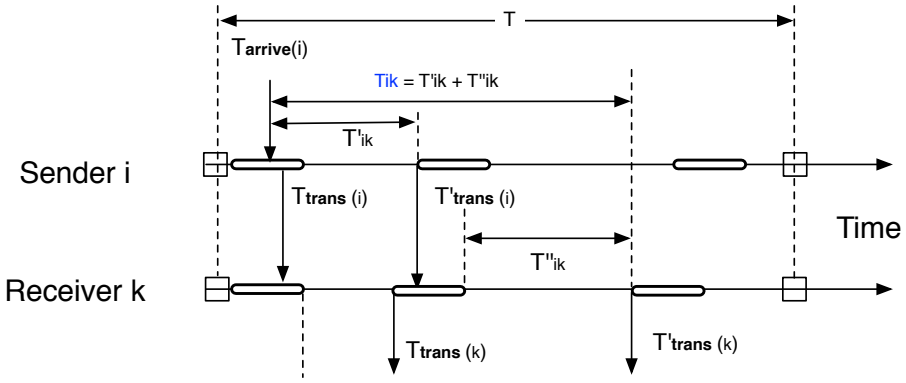


Figure 5.1 Example of transmission latency within one hop.

One-hop latency is evaluated as: $T_{ik} = \alpha_1 \cdot T'_{ik} + \alpha_2 \cdot T''_{ik}$, where α_1 and α_2 are statistical probabilities for patterns of "immediate forwarding" and "hold-and-forward". Here, we select $\alpha_1 = \alpha_2 = \frac{1}{2}$. Thus, L_{ij} is evaluated as follows:

$$L_{ij} = \sum_{l=1}^{m}(P_{ik} \cdot (T_{ii'} + ... + T_{kj})) = \sum_{k=1}^{n} P_{ik} \cdot (T_{ik} + L_{kj}) \tag{5.3}$$

where P_{ik} is packet forward probability for node i to k. T_{ij} indicates the transmission delay from node i to j and the delay inside node j before packet is sent by j. The term of $(T_{ii'} + ... + T_{kj})$ is to sum up all the segments of T_{ik} along path l. Moreover, hitting latency satisfies the recursive condition, hence the estimated latency from each node i to node j can be evaluated by adding local latency to L_{kj}. The packet reachability R_i (estimated number of packet delivery) is evaluated as: $R_i = 1/L_{i,sink}$.

Adaptive Block-based Deconvolution The general principle of deconvolution is described in Section 5.1.2. Particular reasons for adopting degree distribution deconvolution are twofold. *First*, packets initiated with deconvoluted degree distribution have gained recoding opportunities to convolute their degree with other degree distributions in network to achieve final $\mu(*)$. This increases the mixing randomness of data. *Second*, with recoding opportunity, the problem of spatial reuse is addressed.

Though decoding performance is preserved by achieving final degree distribution $\mu(*)$ in decoder, the stopping criteria of each node is no longer the same as $(1+\epsilon)\cdot k$ for successful decoding k samples. Indeed, Raptor Codes support a wide spectrum of output degree function. If employing a modified soliton-like distribution, it requires $(1+\epsilon)\cdot k$ packets to recover k symbols with high probability. If the output degree function is $\Omega(x) = x$, it asks for at least $k\cdot\ln(k/\delta)$ packets for successful decoding with probability $(1-\delta)$. After deconvolution, probability of encoded packet of degree 1 takes major portion. Thus, nodes need send λ times of extra packets. λ is a system parameter of packet redundancy, which is analyzed and evaluated in later section. By considering redundancy parameter λ, the appropriate encoding block can be decided based on the packet reachability $R_i : B_i = \frac{R_i}{(1+\epsilon)\cdot\lambda}$, where ϵ is the Raptor decoding parameter.

Node i, with sensor sampling rate S_i, divides data stream into encoding blocks of size $\hat{B}_i = min\{B_i, S_i\}$. By adaptive block size $\hat{B}_i$, transmissions on node are synchronized with dynamic energy supply underneath. In other words, it can avoid the failure of recovering raw data due to insufficient reception of encoded packets. The rest of nodes $V/\{i\}$ can be considered as an integral node with data size of $S - \hat{B}_i$, in which $S = \sum_{k=1}^{n} \hat{B}_k$. The degree distribution deconvolution can be formulated in following equation:

$$\omega(i) = \begin{cases} \sqrt{\mu(2)}, & i = 1 \\ \frac{\mu(i+1) - \sum_{j=1}^{i-1} \omega(j)\omega(i-j)}{2\omega(1)}, & 1 < i \leq \hat{B}_i \end{cases} \tag{5.4}$$

Notice that the total symbol size S is known as an initial variable for each node. Periodic message of $\hat{B}_i$ from node i is piggybacked in data packet to update S of other nodes.

5.2.3 Opportunistic In-Network Encoding

In the process of opportunistic network recoding, two critical statistical characteristics of erasure codes, namely code degree distribution and randomness in encoded symbol, are well preserved for optimal decoding. First, we exploit "double-deck degree" on encoded packets to ensure that final degree distribution conform with $\mu(*)$. Each node draws local

degree d_i from local distribution $\omega(*)$, as shown in Equation (5.4) and recodes the received data with randomly selected d_i samples. Moreover, an optimal left-distribution is derived in nodes to encode data in such a random manner that data decoding probability is optimized. In the following, we present detailed discussions on the behaviors of both packets and nodes respectively during opportunistic in-network encoding.

Packet coding behavior We adopt a pre-coding LDPC with rate R, and generate $(\frac{1}{R} - 1) \cdot S$ parity check packets. Encoding packet is generated by each sensing node, with a degree d_p associated with the packet. This degree terminates the encoding procedure when it is satisfied. Then packets are forwarded to sink with no further recoding afterwards, so that the degree distribution can be closely approximated. Besides, the node elected as parity check node will associate another degree d_r with packet, which controls the generation of parity check data. Parity-check data will be encoded along with data encoding, but with different randomly selected sets.

As illustrated in Fig. 5.2, d_i is randomly drawn from derived distribution $\omega(d)$. As long as $(c_p + d_i)$ does not exceed the threshold of d_p, d_i samples are encoded; otherwise, the overflowed data are discarded in order to satisfy the degree d_p. Packet $\hat{p}$ is generated to carry both parity-check data and encoded data. We discuss two cases of in-network recoding as follows.

***Case I*:** *carry encoded data only.* When packet $\hat{p}$ originates from a non-parity-check node, the packet has only one associated degree d_p and a counter c_p respectively. It first encodes d_i randomly selected data in starting node, and then be forwarded to neighbor nodes. Whenever d_p is satisfied, packet $\hat{p}$ is marked as "non-recodable" and forwarded to sink with no further recoding.

***Case II*:** *carry extra parity-check data.* In this case, the packet $\hat{p}$ has two degrees, saying d_p (degree for encoded data) and d_r (degree for parity-check data). We initialize two counter c_p and c_r to trace coding vector size. Incrementally, c_p and c_r are updated by adding d_i every time the XOR happens for encoded data and parity-check data. Note that d_i data

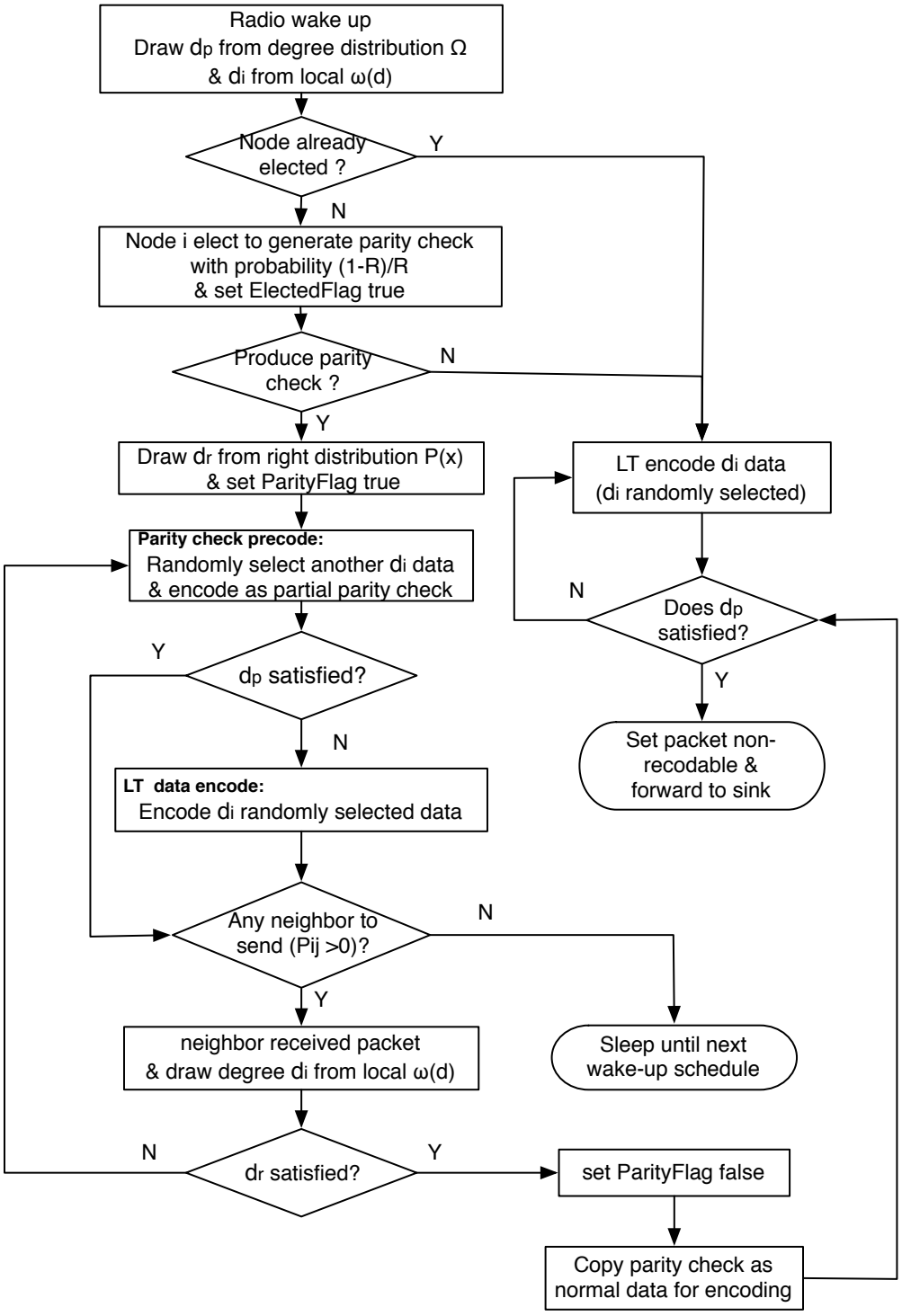


Figure 5.2 Operation flowchart for opportunistic in-network recoding.

set is independently randomly selected for data encoding and parity-check encoding, which are of no correlation.

Node coding behavior Node u has probability $(1-R)/R$ to become a parity-check node. Once node u is elected as the parity-check node, it draws a degree d_p from right probability distribution $P(x)$ of LDPC. Upon receiving packet $\hat{p}$ from its neighbor, node u adds d_i parity-check coding data from local sample pool. Thus, $(\frac{1}{R}-1)\cdot S$ parity-check data can be generated and stored in multiple random walks.

A left probability distribution $\psi(*)$ is derived in each node, satisfying that

$$\Psi(1-\Omega(1-x)) < x \tag{5.5}$$

where $\Psi(x) = \sum_{i=1} \psi(i) \cdot x^{i-1}$ and $\Omega(x) = \sum_{i=1} \omega(i) \cdot x^{i-1}$. Each node uniformly draws a degree d_l from $\psi(*)$ at random. During encoding, each node makes sure that the number of encoded packets, where native data from a source node u is included, is upper bounded by d_l derived in node u.

5.2.4 Algorithm Analysis

In this section, we theoretically analyze the data throughput and communication overhead of *OnCode* under disruptive network conditions, and compare its analysis results with other state-of-art data collection protocols: *NoCode* (custody data delivery), *ONEC* (opportunistic network erasure codes) [36] and *SlideOR* (sliding window based random linear coding) [34]. Given a network of size n, each node's activity is synchronized with its energy dynamics. Nodes' operations (ON/OFF) have to be synchronized with energy supply. The OFF state of node causes node to fail to respond data transmission or reception, which is denoted as node failure. For ease of presentation, we assume node disruption ratio is homogeneous across network, denoted as w. All nodes are data sources which have the certain amount of data needed to be delivered to sink node, which add up to m in total. Then, an

average hop counts for m data is estimated by:

$$E_h = \sum_{i=1}^{\infty} \frac{1}{2^i} \cdot \log_2 \frac{n}{2^{i-1}} = \log_2 n - 1$$

Theorem 21 *Given a network of size n and m total amount of data, node disruption ratio is denoted as w. The delivery ratio of OnCode algorithm is lower bounded by $(1 - \frac{\delta}{e^2})$, with communication overhead as $(m\delta)E_h$, where e is natural number, $\delta = \frac{1}{D}[1-(1-w)^2]$, and D is the expected node degree and $E_h = \log_2 n - 1$.*

Proof 22 *OnCode satisfies the code degree distribution $\omega(x)$ by "double-deck degree", so that the data set can be recovered with high probability if network data encoding procedure is conducted in a pure random manner. Due to the nature of opportunistic network encoding, pure randomness is not guaranteed, affecting the final data delivery ratio. Thus, we derive the delivery ratio by directly analyzing decoding probability given network encoding operation and network data loss rate δ.*

Let $(\psi(1), \psi(2), ..., \psi(L))$ and $(\omega(1), \omega(2), ..., \omega(R))$ be the probability vectors, denoting that each node on the left of encoding bipartite graph has degree i with probability $\psi(i)$, and each node has degree j with probability $\omega(j)$ independently. Then we construct "And-Or" tree [65] from encoding bipartite graph to analyze the decoding probability as follows. Randomly choose one edge (u, v), and construct the subgraph G_s by selecting $s-$hop neighbors of node u and removing all other edges. The data from node u is labeled with δ probability initially, denoting that δ fraction of missing data. The goal of encoding and decoding procedure is to reduce the missing fraction as much as possible. It is clear that data of node u will be recovered if it is labeled as 1 eventually on subgraph G_s.

And data from node u is eventually recovered only if it is directly delivered or at least one of its right-side neighbor v (encoded packet) has all its neighbors decoded except u. Note that node u has i children with probability $\psi(i)$ and node v has j children with probability $\omega(j)$. Thus, from Lemma 1 in [65], we have $y_l = \delta \cdot \Psi(1-\Omega(1-y_{l-1}))$, where $\Psi(x) = \sum_{i=1} \psi(i) \cdot x^{i-1}$

and $\Omega(x) = \sum_{i=1} \omega(i) \cdot x^{i-1}$. *In this recursion, we let* $y_0 = \delta$, y_l *will determine the final recover probability of node* u *in subgraph* G_s. *If the condition* $y_l < y_{l-1}$ *holds, the* y_l *goes to* 0 *as* l *grows. According to Equation (5.5),* $y_l = \delta \cdot \Psi(1 - \Omega(1 - y_{l-1})) < y_{l-1}$. *We set* l *to* $2/\epsilon$, *then it is clear that* $y_l \leq \delta/e^2$, *which means recovery fraction* $(1 - \delta) \geq (1 - \delta/e^2)$.

OnCode adapts source encoding and sending rate, so that m *amount of data can be delivered to destination through opportunistic encoding paths in a disruptive network over estimated (*$\log_2 n - 1$*) hops. Since opportunistic network coding help regenerate loss encoded packets during network encoding, the data loss probability for OnCode is upper bounded by* $\delta \leq \frac{1}{D}[1 - (1 - w)^2]$. *Therefore, the communication overhead to achieve* $(1 - \frac{\delta}{e^2})$ *delivery ratio, is upper bounded by:* $\sum_{i=1}^{\log_2 n - 1} m\delta = m(\log_2 n - 1) \cdot \frac{1}{D}[1 - (1 - w)^2]$.

For comparison, we analyze communication overhead of existing data collection protocols: *NoCode*, *SlideOR* and *ONEC*. *NoCode* is a data collection approach which leverage custody transmission to ensure data reliability. In *NoCode*, the acknowledge control mechanism is carried out within one hop transmission. Sender advance sending window once it receives acknowledges from receiver. The packet loss effect is restricted within one hop distance. In contrast, "end-to-end" TCP reliable data transmission employs cascaded ACK relay from destination to source, and incur much more communication overhead to reach the same delivery ratio in lossy network condition. Without control mechanism, UDP even can not guarantee any delivery ratio. Therefore, we chose to analyze custody TCP transmission denoted as *NoCode* mechanism, which is the best of those no coding approaches.

Theorem 23 *Employing custody based NoCode to deliver* m *amount of data over network of size* n, *it achieves the delivery ratio of* $(1 - \frac{\delta}{e^2})$ *with communication overhead:* $\frac{1}{2}m(\log_2 n - 1)\sigma[1 + \sigma^{\frac{\log_2 \delta/e^2}{\log_2 \sigma}}] \cdot \frac{\log_2 \delta/e^2}{\log_2 \sigma}$, *where* w *is the node disruption ratio (*$0 < w < 1$*), e is the natural number,* $\delta = \frac{1}{D}[1 - (1 - w)^2]$ *and* $\sigma = 1 - (1 - w)^4$.

Proof 24 *NoCode uses custody transfer to collect data over multi-hop networks. ACK control messages are utilized within one hop to ensure the reliable data transmission. Each lost data packet will incur ACK feedback control to notify sender for retransmission. After the*

first data transmission, $m(1-w)^2$ out of m amount of data arrives at receiver, where w is the node disruption ratio. And there are $m(1-(1-w)^2)$ data is lost during transmission, which should be notified by ACK control messages. Note that the ACK messages can also become lost over this lossy communication pairs. After a round trip transmission, the receiver can expect the lost data to be decreased to: $m[1-(1-w)^2]\cdot[1-(1-w)^4]$. After k times of round trip transmission, the expected lost packets is reduced to: $m[1-(1-w)^2]\cdot[1-(1-w)^4]^k$. By constraining expected packet lost less than $\delta = \frac{1}{D}\cdot[1-(1-w)^2]$, we get $k \geq \log_2 \frac{\delta}{e^2} / \log_2[1-(1-w)^4]$. That is to say, the data loss will be no greater than δ, if at least NoCode carries k times of retransmission in every hop. Therefore, the communication overhead inside each hop would be $\frac{m}{2}[1-(1-w)^4][1+(1-(1-w)^4)^k]k$. And in total, the communication overhead is obtained as: $\frac{m}{2}(\log_2 n-1)\sigma[1+\sigma^{\frac{\log_2 \delta/e^2}{\log_2 \sigma}}]\cdot\frac{\log_2 \delta/e^2}{\log_2 \sigma}$, where m is the size of data set, n is the network size, $\delta = \frac{1}{D}[1-(1-w)^2]$ and $\sigma = 1-(1-w)^4$ and $0 < w < 1$.

Theorem 25 *ONEC achieves the delivery ratio of $(1-\frac{\delta}{e^2})$ with communication overhead $m(\log_2 n-1)\cdot\frac{1}{D}[1-(1-w)^2]+\frac{n}{2(1-w)^2}$, where m is the size of data set, n is network size, D is expected node degree, w is node disruption ratio, and $\delta = \frac{1}{D}[1-(1-w)^2]$.*

Proof 26 *ONEC uses erasure codes based coding mechanism, which achieves decoding probability $(1-\frac{\delta}{e^2})$, with δ percentage of lost encoded packets. Its communication cost on delivering raw data is the same as that of OnCode. The difference is that ONEC requires explicit ACK feedback message from destination to source to move to next data block. These ACK message transmission is extra communication overhead compared with OnCode. Retransmission of ACK messages are required to achieve reliable feedback control. Since ACK message is broadcast to all the nodes in the network, and the ACK is retransmitted by intermediate nodes, the total number of retransmission of ACK message is estimated as: $\frac{1}{(1-w)^2}$. And there are only $\frac{n}{2}$ nodes that need to transmit the ACK. Therefore, total communication overhead is: $m(\log_2 n-1)\cdot\frac{1}{D}[1-(1-w)^2]+\frac{n}{2(1-w)^2}$.*

We analyze the delivery ratio and communication overhead of a network coding scheme based on random linear codes (*SlideOR*).

Theorem 27 *SlideOR achieves an upper bound of delivery ratio as* $(1-\frac{\delta}{e^2})$ *with communication overhead* $m(\log_2 n-1)\cdot\frac{1}{D}[1-(1-w)^2]+\frac{m(\log_2 n-1)}{(1-w)^{2(\log_2 n-1)}}$, *where* m *is the size of data set,* n *is network size,* D *is expected node degree,* w *is node disruption ratio and* $\delta=\frac{1}{D}[1-(1-w)^2]$.

Proof 28 *In SlideOR, a coded packet is generated as* $p_j=\sum_i C_{ji}p_i$, *where* C_{ji} *is random coefficient variable selected from Galois Field of size* q, $GF(q)$. *The native data set of* $\{p_1,...,p_N\}$ *can only be recovered when at least* N *linearly independent coded packets are received for decoding. In order to obtain the probability that receiving matrix* G_q *has a full rank* N, *we analyze the coding vector generation. In the first extraction, any non-zero vector* $\{C_{j1},...,C_{jN}\}$ *is a valid coding vector. And the probability for generating such a non-zero vector is* $(1-1/q^N)$. *For second coding vector, there are* q *vectors which are dependent; and* q^2 *vectors dependent with previously generated vectors for the third extraction and so forth. Thus, the probability of having* N *linearly independent coding vector is given by:* $\prod_{j=0}^{N-1}(1-q^j/q^N)=\prod_{j=1}^{N}(1-1/q^j)=(1-1/q)$. *So, by letting* $1-1/q=1-\delta/e^2$, *we obtain* $q=e^2/\delta$. *That is to say that SlideOR reaches a delivery ratio of* $1-\frac{\delta}{e^2}$, *by letting* $q=e^2/\delta$.

For communication overhead, by opportunistic random linear encoding, senders transmits $m(\log_2 n-1)\cdot\frac{1}{D}[1-(1-w)^2]$ *in order to make the receiver obtain at least* $m\cdot\delta$ *encoded packets. This communication cost is the same as ONEC and OnCode. However, receiver apply Gaussian Elimination for decoding and ACK message is initiated immediately to notify sender to advance the sending window. Thus, receiver unicast* m *ACK messages to each of* n *senders to advance sending window. The total communication cost for delivering* m *ACK messages to network nodes is computed as:* $\frac{m(\log_2 n-1)}{(1-w)^{2(\log_2 n-1)}}$, *where* $\log_2 n-1$ *is the estimated hop counts ACK messages is forwarded through. Therefore, the total communication overhead is:* $m(\log_2 n-1)\cdot\frac{1}{D}[1-(1-w)^2]+\frac{m(\log_2 n-1)}{(1-w)^{2(\log_2 n-1)}}$.

Table 5.2 shows the communication overhead in the theorem, where $\sigma=1-(1-w)^4$, $\delta=\frac{1}{D}[1-(1-w)^2]$. From table 5.2, we can observe that under the same network model of size n, data set size m and node disruption ratio w, coding based approaches consume less communication overhead than NoCode approach. And because the ACK in *ONEC* is

Table 5.2 Analytical comparison of communication overheads

protocol	communication overhead	delivery ratio
OnCode	$\frac{m}{D}(\log_2 n - 1)[1-(1-w)^2]$	$(1-\delta/e^2)$
NoCode	$\frac{m\sigma}{2}(\log_2 n - 1)[1 + \sigma^{\frac{\log_2(\delta/e^2)}{\log_2 \sigma}}] \cdot \frac{\log_2(\delta/e^2)}{\log_2 \sigma}$	$(1-\delta/e^2)$
ONEC	$\frac{m}{D}(\log_2 n - 1)[1-(1-w)^2] + \frac{n}{2(1-w)^2}$	$(1-\delta/e^2)$
SlideOR	$\frac{m}{D}(\log_2 n - 1)[1-(1-w)^2] + \frac{m(\log_2 n - 1)}{(1-w)^{2(\log_2 n - 1)}}$	$(1-\delta/e^2)$

broadcast to network nodes instead of unicast to each node, *ONEC* has less communication overhead than that of *SlideOR*, where sliding window in sender side only advances when it receives unicast ACK feedback from receiver. However, *OnCode* moves to next data block without the ACK notification, significantly reducing the communication overhead to only $\frac{m}{D}(\log_2 n - 1)[1-(1-w)^2]$.

This analytical results are illustrated in Figure 5.3, where we let $m = 500$, $n = 100$, $D = 5$ and $e = 2.718$. It is observed from Figure 5.3 that communication overhead of *NoCode* approach increases exponentially when node disruption ratio grows. It is because both hop by hop ACK messages and the message retransmission explode when nodes become more disruptive from the network. While *ONEC* uses opportunistic in-network coding, it still requires feedback control to move to next data block. Thus, *ONEC* consumes more message overhead than *OnCode*. But the feedback message *ONEC* needs only broadcast instead of unicast, so it still consumes much less than *SlideOR* which applies the unicast for each of feedback control.

5.3 System Design and Implementation

In this section, we describe the design of energy-synchronized component and protocol implementation details of OnCode protocol in TinyOS [71].

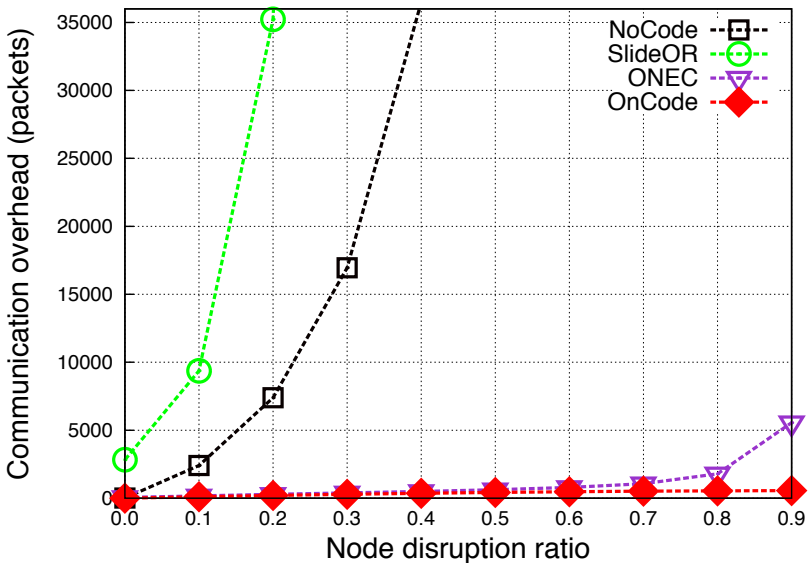


Figure 5.3 Illustration of theoretical analysis

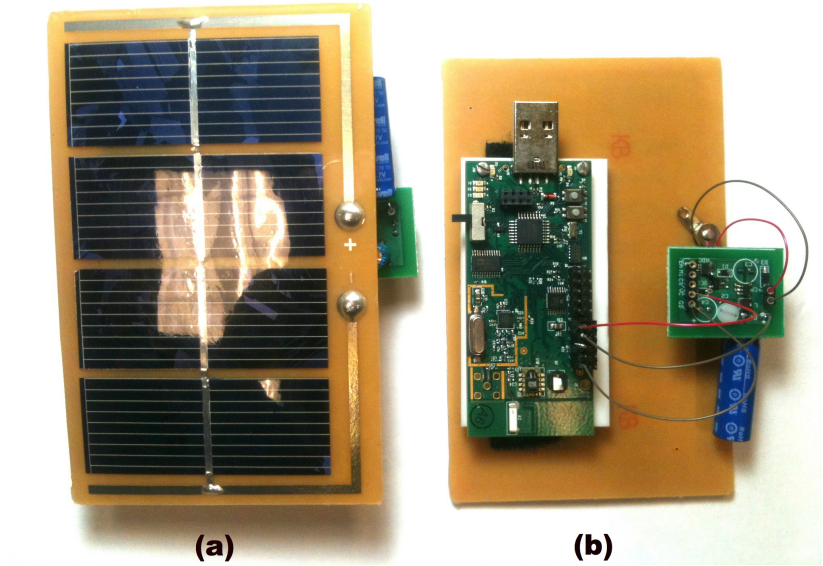


Figure 5.4 Energy-Synchronization Module. (a) Front View: Solar Cell; (b) Rear View: TelosW mote powered by PMS with ultra-capacitor as energy storage unit.

5.3.1 Energy-Synchronization Module

As illustrated in Fig. 5.4, our Energy-Synchronization Model (ESM) consists of solar cell, Power Management System (PMS) and TelosW sensor mote. Solar cell is used in our system, which can be charged by either fluorescent or incandescent light indoor, with an open-circuit voltage of $2.0V$. PMS carries an ultra-capacitor of $10F$ as energy storage, which holds a voltage of $1.8V$ when fully charged. PMS regulates its output voltage to $3.3V$ as the voltage of ultra-capacitor reaches over $0.6V$. TelosW sensor mote discharges the energy from PMS. ESM keeps track of the energy gain and consumption in past time slots, conducts prediction for the future time slot, and adapts the wake-on ratio of TelosW mote. The functionalities of ESM are elaborated as follows.

First, ESM evaluates energy gain and consumption on time slot basis. For clear expression, we denote the energy harvested in time period $\Delta t = (t2 - t1)$ as $E_h(\Delta t)$, and the energy consumption as $E_c(\Delta t)$ for the same time slot. The energy consumption $E_c(\Delta t)$ is the difference of energy meter values at two time points: $e_t(t2) - e_t(t1)$. Since $E_c(\Delta t)$ is obtained from on-board energy meter, the value corresponds to the total current drawn by TelosW mote. For computing the energy harvested by solar cell $E_h(\Delta t)$, we adopt equation: $E_h(\Delta t) = E_{lk} + E_c(\Delta t) + E_{cap}(\Delta t)$. The leftover energy change in capacitor has the relations to the voltage change of capacitor V_{cap} as: $E_{cap}(\Delta t) = \frac{1}{2}CV_{cap}^2(\Delta t)$. And E_{lk} is the leaking energy in capacitor, which carries in a rate related to the voltage. ESM obtains E_{lk} from empirical data.

Second, ESM predicts the energy gain in the future slot, $\hat{E}_h(t)$, based on estimated energy gains. Exponentially Weighted Moving Average (EWMA) is applied to estimate the $\hat{E}_h(t)$ by applying weight factors. We estimate $\hat{E}_h(t)$ as a weighted sum of the observed harvested energy as: $\hat{E}_h(t) = \alpha \cdot \hat{E}_h(t-T) + \alpha \cdot (1-\alpha) \cdot \hat{E}_h(t-2T) + ... + \alpha \cdot (1-\alpha)^k \cdot \hat{E}_h(t-(k+1)T)$. The coefficient α represents the degree of weighting decrease, a constant smoothing factor between 0 and 1. Based on predicted value of $\hat{E}_h(t)$, ESM tune wake-on ratio to satisfy condition: $E_c(t) + E_{lk} \leq \hat{E}_h(t)$.

Third, the prediction may result in a bad performance due to dramatic energy fluc-

tuations. Thereafter, besides energy prediction, Additive Increase Multiplicative Decrease (AIMD) approach is adopted to adjust wake-on ratio in order to react to the energy gain changes rapidly. Prediction-based and reaction-based adjustment work in concert to fulfill the goal of energy synchronization.

5.3.2 Protocol Implementation

The software protocols are developed and implemented in TinyOS [71]. The protocol stack is illustrated in Fig. 5.5, and shaded parts are our contributions of "OnCode Protocol" and "Energy Synchronization" respectively.

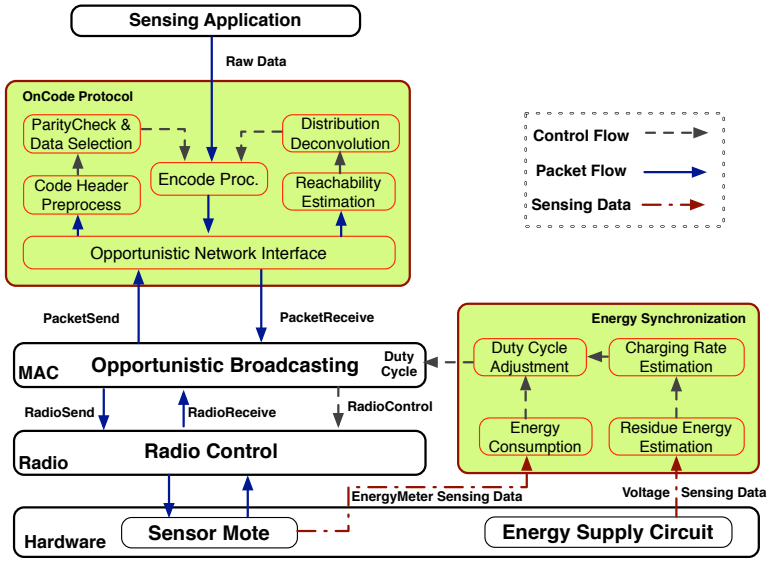


Figure 5.5 Protocol Implementation.

OnCode implementation comprises of following components:

1. *Energy-Synchronization Module* estimates energy gain and consumption, and tunes wake-on duration dynamically.

2. *Reachability Estimate* is to initialize and update the reachability of node to sink based on wake-on ratio. It is to be noted that, each node only needs to know the duration ratio of ON and OFF mode of neighbor nodes, and do not need to know the exact time slots of ON and OFF.

3. *Distribution Deconvolution* is to adaptively determine the block size for encoding and conduct deconvolution of degree distribution correspondingly.

4. *Opportunistic In-Network Encode* is in charge of encoding raw data and parity check data with randomly selected symbols.

5.4 Experimental Evaluation

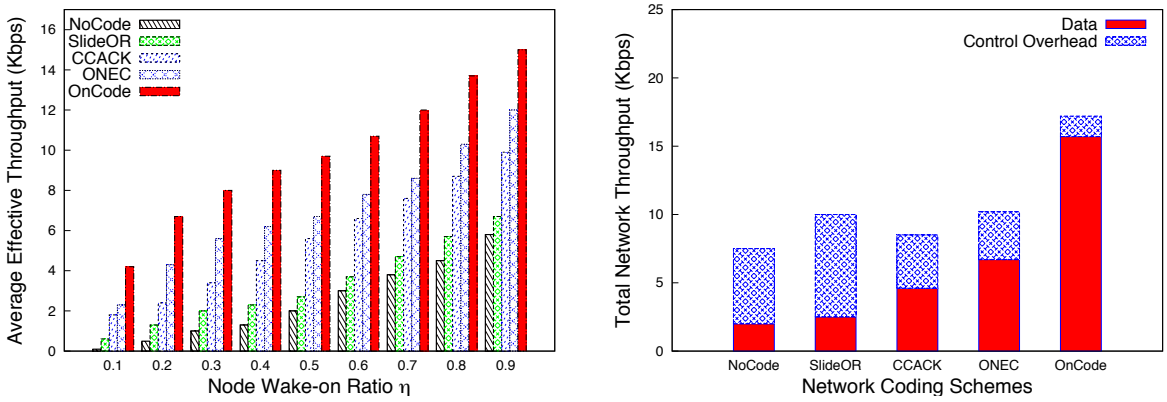
We conduct the simulation in Section 5.4.1 first to evaluate *OnCode* performance with two purposes: *first*, the efficiency of *OnCode* algorithm can be systematically studied by tuning the system parameters in the simulation; *second*, the scalability of algorithm can be extensively evaluated in simulation environment. The *OnCode* algorithm is further validated in a real energy-synchronized test-bed in Section 5.4.2.

5.4.1 Simulation Evaluation

The simulation experiment is conducted in TOSSIM [69]. In the simulation, a network of 100 nodes is deployed in $100m * 100m$ area at random. Only one sink node is present in the network to decode and collect data. Each node sends sensing data to sink node, subject to the energy supply in each node. Energy distribution is simulated by different input profiles.

We evaluate the average data throughput of OnCode by varying energy distribution, packet redundant ratio λ, and forwarding probability ρ. Additionally, the communication overhead is evaluated as well. All the performances of OnCode are compared with four other coding schemes: *NoCode*: It does not employ coding schemes in data delivery; *ONEC* [36]: A tree structure based degree distribution deconvolution, and opportunistic recoding is applied; *CCACK* [5]: A node can suppress redundant packets of its neighbors, by broadcasting the acknowledge message; *SlideOR* [34]: SlideOR applies random linear codes and utilizes the sliding window to encode intra-flow data traffic. The result is the average of 500 rounds of simulation runs.

Impact of varying energy distribution Since communication operations must be synchronized with available energy to fulfill the lifetime requirement, wake-on ratio η of nodes are derived by ESM according to the energy supply. First, we assign the homogeneous energy to network nodes with varying amount, resulting in wake-on ratio η from 0.1 to 0.9, as illustrated in Figure 5.6(a). The average effective throughput of OnCode increases from $4Kbps$ to $14Kbps$.



(a) Impact of system parameter η (b) communication overhead

Figure 5.6 Impact of system parameter η and Communication overhead.

In Figure 5.6(a), OnCode outperforms other data delivery approaches in varying wake-on ratio η. It is due to following reasons. *First*, the sole reliance of ACK to advance to next encoding window is alleviated in OnCode. *Second*, OnCode takes advantages of different wake-on ratios in the opportunistic delivery paths, by accumulative packet reachability R_i. The encoding window and degree distribution are adaptively tuned to improve the data delivery throughput. *Third*, the protocol requires reduced control overhead, and updated wake-on ratio η from neighbor nodes can be piggybacked in the data packets.

We also evaluate impact of heterogeneous energy distribution on effective data throughput. In Figure 5.7, it shows different distributions of energy profiles, normalized by $10mW$. In general, effective network throughput increases as average energy supply rises. Moreover, the energy map shows that average effective throughput increases when the gradient of energy distribution increases towards sink, as illustrated in "Near-Sink" of Figure 5.7 com-

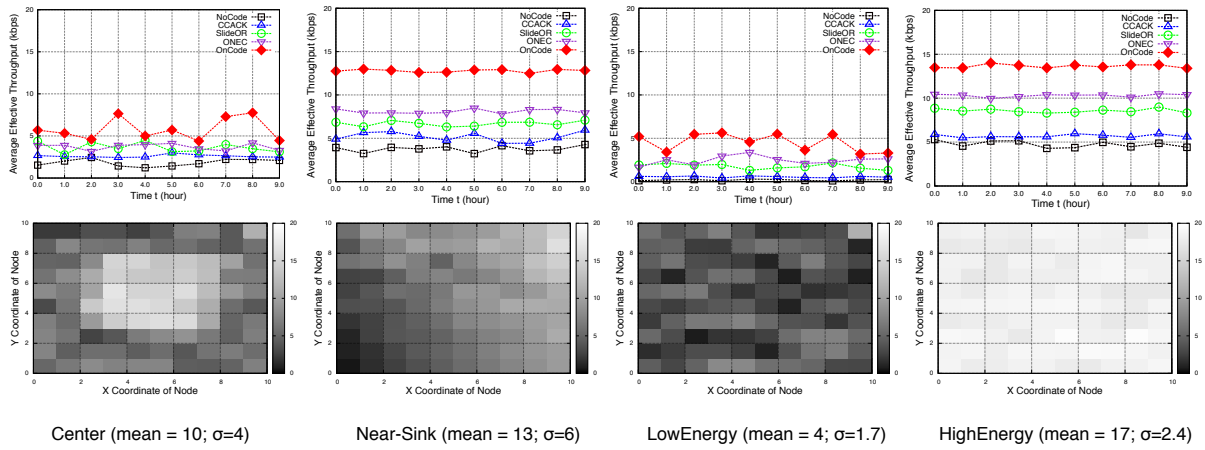


Figure 5.7 Average effective throughput under different energy maps. Each block indicates the energy in each node, and different gray scales denote energy levels. Energy is in the unit of $10mW$, "mean" is the mean energy value across network, and σ is the energy deviation.

pared with "center". With the average energy $17 * 10 = 170mW$ and deviation of $24mW$ for communication in the case of "HighEnergy", mean effective throughput reaches $14kbps$ for OnCode. Other coding schemes have less effective throughput, although they all experience certain increments on throughput.

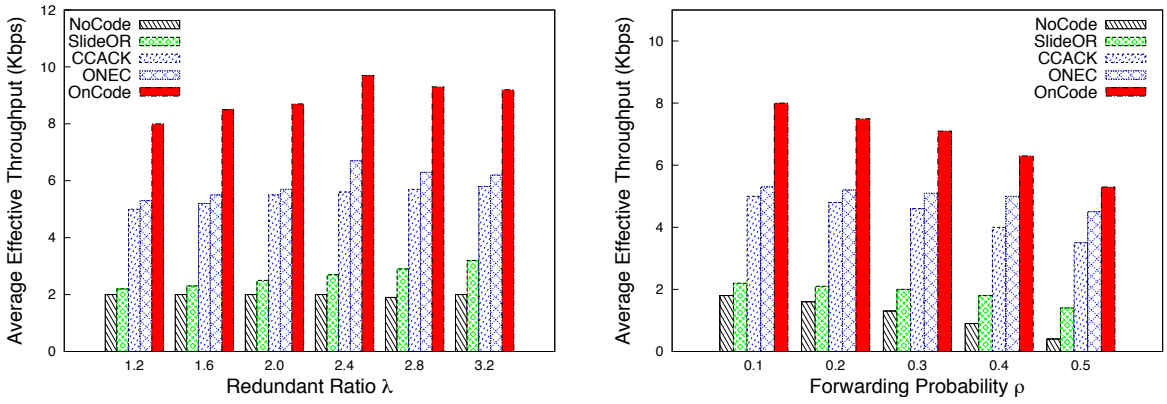
Communication Overhead We evaluate communication overhead of OnCode protocol and compare with the other approaches. The dedicated communication overhead in OnCode includes broadcast message to notify initial neighbors' wake-on ratio and broadcasting messages of updated global symbols. Parity check bits and updated wake-on ratio from neighbors are piggybacked in the data packet. Though they make much less radio overhead compared to dedicated control message, we still count them.

Figure 5.6(b) illustrates all the composition of entire throughput. The overhead in OnCode is less than other coding schemes. NoCodes still requires ACK feedback, which are considered as control message as well. If considering the ratio between control overhead and effective data throughput, OnCode presents a much smaller portion of cost than others. It is attributed to the reason that OnCode is a structure-less routing protocol, in which coding and forwarding packets follows probabilistic delivery paths with less deterministic control

messages.

Impact of varying packet redundant ratio λ The role of redundant packet ratio, λ, is to decide the ratio of packets delivered to the number of packets required for decoder for successful decoding, based on packet reachability estimation. The redundant packets are injected by source to combat the dynamic network fluctuations. An explicit trade-off is that the increasing redundant ratio λ can enhance the decoding probability, hence improving the effective throughput. On the other hand, the extra packets can also be the useless communication overhead which does not help in decoding.

In Figure 5.8 (a), redundant ratio λ increases from 1.2 to 3.2. λ has less effect on SlideOR, CCACK and NoCode. Redundant packet ratio of 2.4 explores the maximum trade-off in OnCode and ONEC codings. This is attributed to the fact that both of them adopt network erasure encoding, which applies similar Belief Propagation algorithm to decode packets.



(a) λ (b) ρ

Figure 5.8 Impact of system parameter λ and ρ.

Impact of varying forwarding probability ρ System parameter ρ influences the contribution of node priority ranking to the packet forwarding probability, and the network throughput. We vary parameter ρ from 0.1 to 0.5. As illustrated in Figure 5.8 (b), as ρ gets larger, the average effective throughput reduces. It is due to the fact that when ρ increases,

neighbor nodes have less forwarding probability. With smaller ρ, neighbor nodes with lower priorities can have more forwarding probabilities, which indicates packets can take advantage of more spatial diversity.

5.4.2 Testbed Evaluation

We present the proof-of-concept of OnCode protocol, and validate that OnCode can achieve good quality of data delivery over energy-synchronized sensor network. Sixteen TelosW [64] nodes are used in the experiments, and they are driven by ESM system, as shown in Fig. 5.4. Sensor nodes are deployed to form line topology with sink node in one end, or grid topology with sink node in the top left corner. The radio power level of $CC1101$ is set to the lowest level as $-5dBm$ to construct multi-hop communication, with 2.5 meter physical distance between nodes. Sink node is connected to and powered by PC gateway, so that it does not perform energy management and wake-on schedule.

A small capacitor of $10F$ is adopted in ESM for indoor environment. We synthesize the light intensity variation, by performing different temporal patterns of on and off in lamp light. We obtain the energy leakage rate of ultra-capacitor during empirical study and incorporate into the energy consumption to calculate the energy gain from environmental light. Experimentally we adopt $\alpha = 0.8$ to make the EWMA energy prediction adapt to energy dynamics fast. Every ESM starts with fully charged capacitor.

Energy-Synchronized System Sustainability The design goal of ESM system is to tune node's wake-on ratio, so as to prevent sensor node from running out of energy supply and ideally perform perpetually. However, the indoor lighting condition can be worse than solar radiation over the day time. Our dynamic pattern of lighting intensity pose even more severe challenge, as shown in Fig. 5.9.

This experiment is conducted for 12 hours to evaluate the system sustainability under dynamic energy recharging condition. Every 100 minutes, renewable energy sources are present for harvesting, and the duration also vary over time. Highest peak of recharging

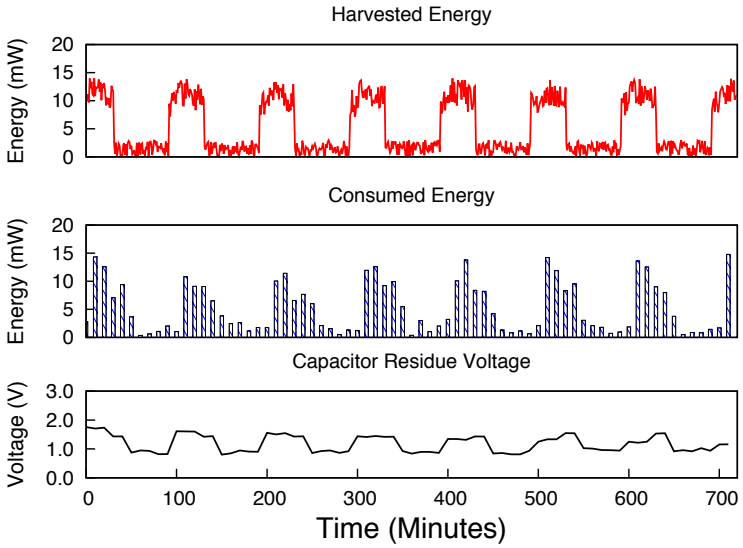


Figure 5.9 Energy trace: (top) trace of 12-hr harvested energy in mW, (middle) node energy consumption, and (bottom) residual voltage of ultra-capacitor

rate can reach $14mW$, while it can drop to $3mW$ within 10 minutes. ESM reacts to this dramatic change by multiplicatively decreasing the wake-on ratio after detecting the decline in capacitor voltage. Thus, the falling of residual voltage of capacitor is prevented in a timely manner. For the first 7 hours, the average capacitor voltage declines gradually even under short-term recovery. ESM compensates this long-term loss by automatically further reducing the wake-on ratio, leading to a self-sustainable energy-synchronized system.

Network Data Throughput Network throughput is evaluated in the testbed of 16 nodes deployed in grid and line topology respectively.

In Fig. 5.10, with the same pattern of varying energy recharging rate, there are distinguishable observations between two topologies: (1) the average network throughput in grid topology is 40% higher than that of line topology. Line topology of maximum 15 hops is much longer than maximum 6 hops in grid topology, and there is fewer forwarding opportunities in line topology. Larger opportunistic forwarding probability and encoding choices in grid topology result in a lesser overhead and higher decoding ratio than in line topology. (2) In grid topology, the surge of energy recharging rate enhance network throughput in a

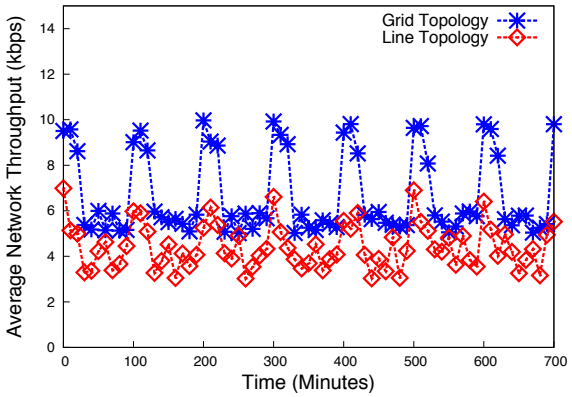
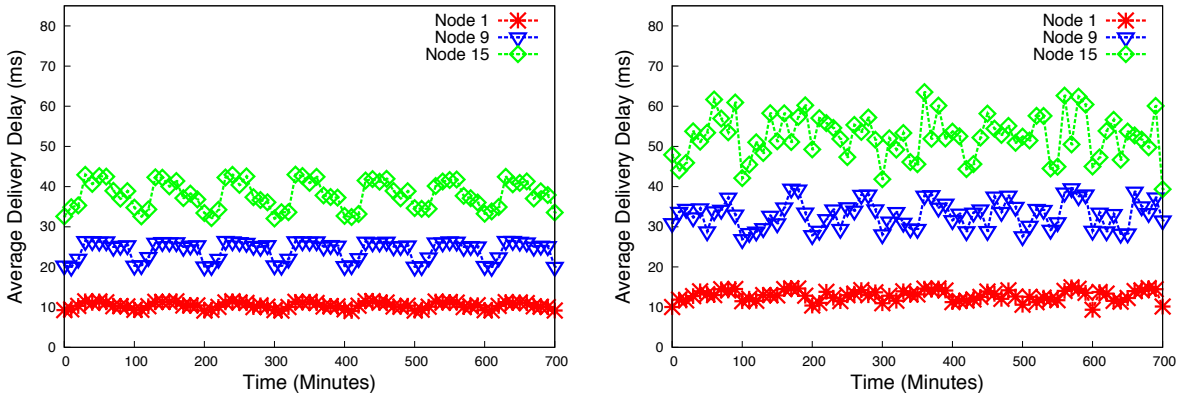


Figure 5.10 Average throughput over time.

significant way. However, the network throughput of line topology stays insensitive to energy changes. It is also observed in line topology that the energy surge in far-end node has less impact on the throughput enhancement, compared to that in the grid topology.

Network Data Latency Delivery latency is evaluated in grid and line topology respectively. Average delay from node 1, node 9 and node 15 are shown in Fig. 5.11.



(a) Grid Topology

(b) Line Topology

Figure 5.11 Average Delivery Latency.

In grid topology, the average delivery latency is 9 milliseconds per packet for node 1, which is one quarter of the average latency of node 15. It is observed that the latency in node 15 has less fluctuations among nodes in the network due to two factors: (1) packets from node 15 experience less opportunistic forwarding, such that latency deviation may vary

based on the specific travel paths to sink. (2) Recharging energy variation also make the reachability of node 15 to sink change in a less dynamic way. In contrast, in line topology, it costs 25% more on delay for node 15, 11% more on node 8. It is because that in line topology, nodes with multiple hops distance has less opportunistic delivery paths, and hence more delay is introduced to each packet forwarding transmission.

Another observation is that grid topology provides better robustness to temporal energy instability. In other words, the deviation of jitter is less sensitive to the energy fluctuations in grid topology, compared with that of line topology.

5.5 Related Works

The related work presents the state-of-art in energy harvesting and network coding based delivery method respectively.

5.5.1 Micro Solar Power System

Recent research works on micro-solar system have shown the potential of driving low-power devices without battery, such as Prometheus [19] AmbiMax [20], Heliomote [21] solar system, and Trio [22]. The work in [23] proposes a model and guideline for analyzing the design of micro-solar system based on the empirical study of Heliomote and Trio system. The work in [24] proposes a leakage-aware energy control layer to adapt the operation of application to harvesting energy. [25] proposes energy-harvesting low-power device, EnHANTs. [26] study how to allocate energy spending rate with various predictable energy inputs. Our contribution is a coding-based network protocol that can adaptively utilize the energy from micro-solar power system. It is evaluated on a real micro-solar powered indoor testbed.

Based on the solar-powered devices, SolarStore in [27] provides a storage-centric service, which adaptively balances between the data reliability and data sensing. SolarCodes [28] maximize the usage of surplus energy by adjusting the redundant factor of erasure coding in each link respectively. OnCode protocol is distinguishable in two perspectives: (1) OnCode synchronizes data delivery with any energy constraints, not only of surplus energy, but also

of insufficient energy. (2) OnCode exploits opportunistic routing instead of a pre-determined routing, which reduces the coding overhead significantly. The diversity and randomness in opportunistic coding improve quality of data delivery by enhancing flow throughput and fairness.

5.5.2 Network Coding for Data Delivery

The opportunistic routing proposed in ExOR [3] utilizes the probabilistic receiving in multiple hops distance to accelerate the packet forwarding. MORE [4] takes advantages of spatial reuse by random linear coding to improve the data delivery throughput. CCACK [5] utilizes a Null-Space Based (NSB) message to acknowledge the reception of encoded packets, which suppress not only redundant packet retransmission, but also the non-innovative encoded packets. Though CCACK improves MORE by suppressing overhead packet from transmission, the time of moving to next data segment still relies on the ACK from destination. SlideOR in [34] explores sliding window mechanism. But SlideOR is sensitive to ACK notification from decoder to advance encoding window, which would be impractical in disruptive communication environments. ONEC [36] is a recent work to explore network erasure coding in disruptive sensor networks. However, ONEC heavily relies on a strict structure, e.g. tree structure, to conduct recursive deconvolution of degree distribution, which brings considerable overhead when updating degree distribution during disruption. Additionally, the on-path opportunistic recoding may result in distorted final degree distribution which hamper the decoding success probability. ONEC is not adaptive to the dynamic data rate and energy source. Our technical contributions of OnCode are three-fold: *first*, OnCode conduct deconvolution based on a distributed packet reachability estimation, with low communication overhead when updating. *Second*, OnCode self-tunes the data coding and probabilistic forwarding *in-situ* with energy distribution and variations. *Third*, OnCode maximizes the space diversity and probabilistic packet transmission by opportunistic in-network encoding, while guaranteeing the desired coding degree distribution.

CHAPTER 6

DATA PERSISTENCE WITH STORAGE-CONSTRAINED NETWORK CODING (ECPC)

A sensor network consists of spatially distributed devices communicating with radios and cooperatively sensing physical or environmental conditions. It has found critical applications in catastrophic or emergency scenarios, such as floods, fires, volcanos, battlefields, where human participation is too dangerous and infrastructure networks are impossible or too expensive. However, in those challenging environments, sensor nodes can become unstable and disruptive, even fail occasionally, thus preserving data in a disruptive sensor network is a challenging research subject. Efforts to engage raw data replication for provisioning data persistence is only a makeshift fix, since it is cumbersome and may not scale with the network size. In fact, the constrained sensor memory space and increasing scale of network make the raw data replication inefficient. Some recent works explore the advantage of erasure codes, i.e. LT Codes [17] and Raptor Codes [18]. In [39–41], the sensor nodes disseminate the data to storage nodes through multiple random walks, which aim to simulate the random selection of neighbor symbols during encoding. Each storage node has an associated stop probability for each random walk packet based on the code degree it draw from the degree distribution. Based on random walk theory, a random walk can stabilize after a sufficiently long walk, and the distribution of packet number in storage nodes will conform to the expected code degree distribution. Therefore, original data can be recovered by decoding any sufficient subset of encoded packets from the network. However, the communication overhead during random walk procedure is considerable. In addition, random walk could fail in the middle since node failures could also happen during random walk.

In this paper, a distributed Erasure Coding with randomized Power Control (ECPC) algorithm is proposed. This work makes a distinction from the literature: it achieves data

persistence only by localized data packet broadcasting and erasure coding. In particular, distributed erasure coding procedure of ECPC does not incur multiple random walks, which travel network nodes to ensure the coding randomness and degree distribution, thus resulting in low communication overhead. The basic idea works as follows. Firstly, each node determines the number of encoded packets in storage, which are required to guarantee a successful data recovery with high probability, assuming the failure probability of every node is a given knowledge. If the failure probabilities is heterogeneous, we conservatively use the maximum failure probability of the network as the failure probability for each node, which makes ECPC tolerate the worse case. Second, each node evaluates a posterior probability distribution of radio transmission power, based on which each packet broadcasting is conducted. The required code degree distribution of the chosen erasure codes is proved to be satisfied after sufficient bouts (storage redundancy τ) of broadcast and encoding. The above derivation of posterior distribution of transmission power is only executed occasionally, if the network conditions do not change.

Notice that, the proposed ECPC protocol are generally applicable with any other erasure coding schemes. Given different code degree distribution of different erasure code, we just need to compute the posterior probability distribution of transmission power levels differently. In this paper, we choose Raptor codes [18] as the erasure coding scheme for illustration. Raptor codes, as with fountain codes in general, encode a given message consisting of a number of symbols, k, into a rateless sequence of encoding symbols such that knowledge of any k or more encoding symbols allows the message to be recovered with high probability. Raptor codes are the first known class of fountain codes with linear time encoding and decoding, and are a significant theoretical and practical improvement over LT codes [17].

It is not hard to see our ECPC algorithm has advantages over the random walk based approaches in terms of efficiency and reliability. Firstly, with several localized broadcast instead of several random walks, it considerably saves the communication overhead significantly. Secondly, nodes only have to encode the data received from neighbors, which has a faster termination time. Rapid termination makes our ECPC more reliable in the disruptive

network conditions, since node failures could also fail during the procedure. However, implementing this advantage into a plausible design needs to solve several research challenges: (1) ECPC has to identify the number of encoded packets needed in each individual node, given the node failure probability. A balance between redundancy introduced and desired data persistence is non-trivial. (2) Transmission power of each node needs to be adjusted locally while ensuring the distribution of received packet number conforms to the expected code degree distribution. (3) Pseudo randomness needs to be proved to approximate the original decoding performance of erasure codes, in terms of the number of encoded packet required. We have successfully solved them in this work as presented later. Our theoretical analysis shows that the collaborative storage with the ECPC mechanism can preserve all data with high probability when some nodes fail. In other words, under ϕ percent node failures, all network data can be recovered from nodes with probability $(1-6\cdot\exp(\frac{(1+\epsilon)}{(1-\phi)\cdot\Gamma\cdot 0.06}-8))$, where Γ is the number of encoded packets per node and $0<\epsilon<1$. Notice that Γ is configurable as long as it is larger than $\Gamma_{min}=\frac{1+\epsilon}{(1-\phi)\Lambda}$, where $\Lambda=(0.48+0.06\ln\frac{\delta}{6})$ and $0<\delta<1$. Larger Γ means higher redundancy and higher data recovery ratio. Performance comparisons between ECPC and random walk based approaches show that ECPC mechanism reaches higher data reliability under varying node failure probabilities. In addition, our approach is scalable and has low communication overhead.

6.1 Distributed Erasure Coding with Randomized Power Control

In this section, we give the network model and problem statement, and present the overview of *distributed Erasure Coding with randomized Power Control* (ECPC), followed by a walk-through example.

6.1.1 Network Model and Problem Statement

We model sensor network as a random geometric graph [72], where $|V|$ nodes are uniformly and randomly deployed in a area $\mathcal{A}=[D,D]^2$, and D is the length for each of dimension of the deployed region. Then network density is denoted as: $\eta=n/D^2$. Note

that the area can have different lengths in each of dimensions, we use D for the ease of presentation. We also assume that nodes can broadcast its packet at different power level P_i in the range of $[P_{min}, P_{max}]$. Here, the maximum radio communication radius does not necessarily cover the entire network. The upper and lower bound on transmission power level are constrained by the physical transmission power output. The value can be obtained from datasheet of selected radio type.

Each node senses its surrounding environment and generates data at the same rate. Every data sample is considered as equally important, so that the raw data should be equivalently preserved. Each node is *sensing* data and also *storing* data with its limited memory storage. In addition, nodes inside the network fail with ϕ probability due to exceptional reasons, like environment changes, hazard damage and system crash. The failure event of nodes under consideration is random and independent from each other. This work does not consider spatially correlated failure models.

The research challenge is to preserve sensor data in disruptive sensor network without node repairing. In particular, with ϕ percent fail nodes, the original n data items can be successfully recovered by decoding packets retrieved from available nodes. The repairing of malfunction nodes and corresponding data replacements are out of scope of this paper. The prior knowledge available to each node is limited to network size n and node failure probability ϕ.

The mathematical notation and meaning in ECPC algorithm is shown in Table 7.1.

6.1.2 ECPC In a Nutshell

ECPC utilizes network erasure coding scheme to accomplish distributed data storage in disruptive sensor networks. It works by multiple-round encoding, where for each round a local broadcast is carried out to offload sensing data to other storage nodes. The encoding of ECPC works by XORing data received from multiple neighbors' broadcasts into one packet. The transmission power of each node is locally and randomly adjusted based on a posterior

Table 6.1 Notation in ECPC Algorithm

Notation	Meaning
n	network size $\|V\|$
ϕ	node failure probability
η	network density
P_i	transmission power level
$R(P_i)$	radio transmission range under P_i
$\mathcal{N}_u^i$	neighbor set of u at power level P_i
Γ	number of encoded packets per node
$\Omega(\cdot)$	code degree distribution
d	code degree
$\mathcal{C}_m(\lambda, \rho)$	LDPC parity check coding
$f_P(P_i)$	power distribution function
E	encoded packet
h	range of transmission power set
δ	probability of decoding failure
Λ	ratio on number of packets needed to decode single symbol between Raptor and ECPC

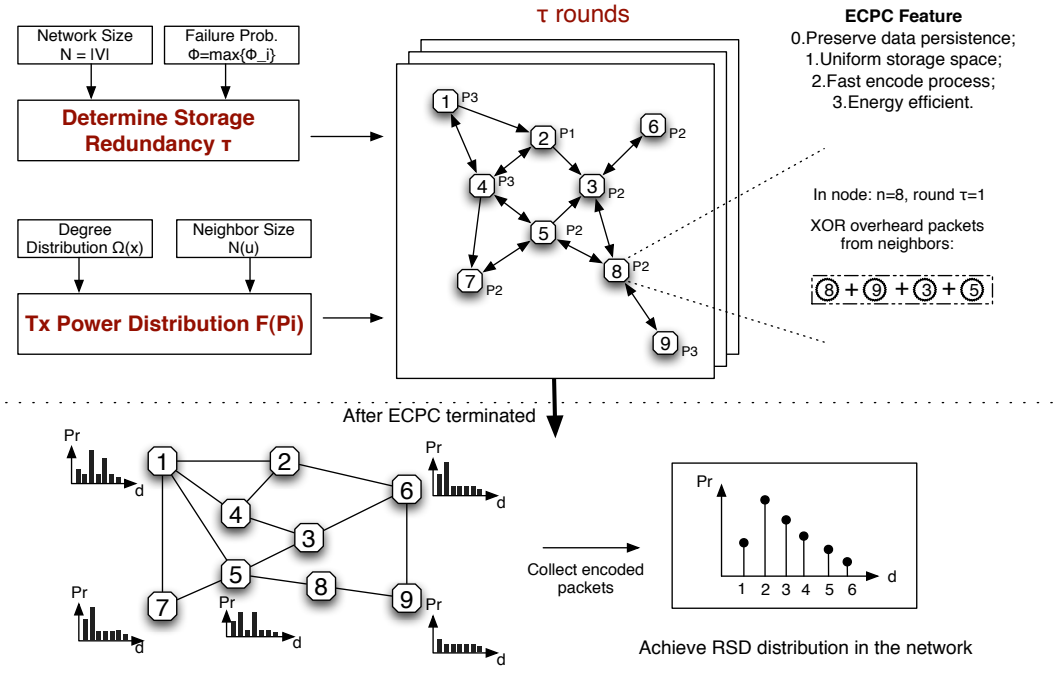


Figure 6.1 ECPC Overview.

Algorithm 6 ECPC Distributed Data Storage

Input: Node failure probability $\phi = max\{\phi_i\}$, final degree distribution $\Omega(*)$ and network size $N = |V|$

Output: Encoded packets stored in distributed nodes: $\mathcal{E} = \bigcup_{E_i}\{\hat{e}_1, \hat{e}_2, ..., \hat{e}_t\}$ ($\forall\ i \in V$ & $\forall\ t \in [1, \Gamma]$)

1: $\mathcal{E} = NULL$;
2: set $\Lambda = (0.48 + 0.06 \ln \frac{\delta}{6})$ (Analysis on randomness in Section 6.2.3);
3: $\Gamma = \frac{1+\epsilon}{(1-\phi)\Lambda}$;
4: **for** $i \in V$ **do**
5: $\quad f_P(P_i) = RPC(\Omega(*), \Gamma, N)$ (Algorithm 7);
6: $\quad E_i = DEC(\Gamma, f_P(P_i))$ (Algorithm 8);
7: **end for**
8: $\mathcal{E} = \bigcup_i\{E_i\}$

probability distribution. Figure 6.1 illustrates the ECPC design in a nutshell.

First, given the node failure probability ϕ and network size N in Figure 6.1, the requisite rounds of encoding, τ, is determined to deliver sufficient amount of encoded packets to storage nodes, ensuring data decoding with high probability. Second, a posterior distribution of transmission power level in node u is derived based on degree distribution $\Omega(x)$ and its neighbor size $N(u)$. Then the transmission power of sensor node is locally adjusted following the derived power distribution (see Section 6.2.1 for details). Sensor nodes repeat broadcasting data using the randomly adjusted transmission power in τ rounds, until the requisite encoded packets are delivered and offloaded. Upon receiving overheard packets, distributed network erasure coding is conducted. In every round, every node encodes received data from neighbors, shown in Figure 6.1 (see Section 6.2.2 for details). Finally, the decoding algorithm based belief propagation is executed to recover data from encoded packets. Notice that the power control algorithm is conducted only when the network starts or network conditions, such as network size, sensing rate or failure probability change dramatically. Otherwise, ECPC only needs to run distributed erasure coding as described in Section 6.2.2.

Algorithm 6 illustrates ECPC for distributed data storage. The output of ECPC algorithm is a set of encoded packets distributed in nodes, i.e. $\mathcal{E}$, whose aggregated degree distribution comply with final degree distribution $\Omega(*)$. In line 2, the decoding coefficient is

assigned based on the randomness analysis in Section 6.2.3. Line 3 determines the storage redundancy τ in each node to achieve $(1-\delta)$ decoding success probability. From line 4 to 7, randomized power control (Algorithm 7) and distributed erasure coding (Algorithm 8) are carried out in tandem with each other.

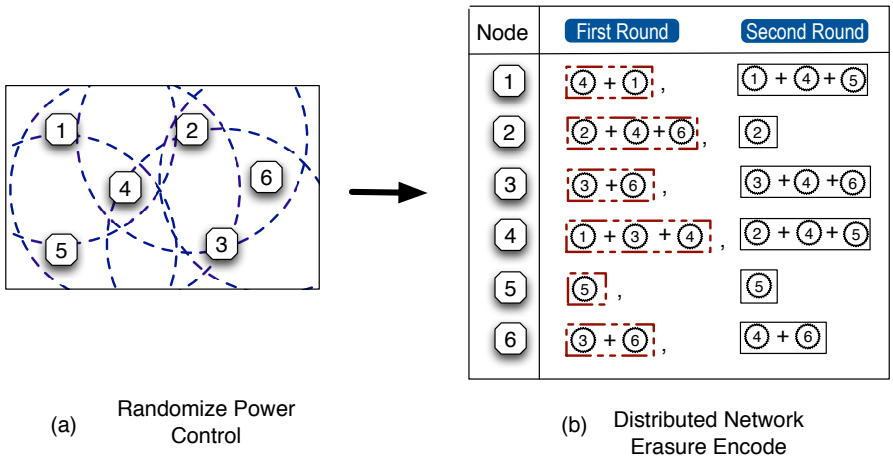


Figure 6.2 ECPC example.

We further illustrate ECPC algorithm by a walk-through example. In Figure 7.1, we assume that the failure probability $\phi = 10\%$, $\delta = 0.1$ and $\epsilon = 0.05$. In the first step, we calculate the requisite amount of encoded packet: $\Gamma = \frac{1+\epsilon}{(1-\phi)\Lambda} = 5$, where $\Lambda = (0.48 + 0.06 \ln \frac{\delta}{6})$. It suggests that each node encode 5 packets through 5 encoding rounds. Due to space constraint, two out of five encoding results are shown in Figure 7.1. For the first round, each node selects a power level randomly from power distribution $f_P(P_i)$. The power assignment in the first round is $\{(1, \underline{99}), (2, \underline{95}), (3, \underline{99}), (4, \underline{103}), (5, \underline{97}), (6, \underline{97})\}$, where the first element is node ID, the second underscored item is the power level randomly drawn. The power range is $[95, 127]$. After sensor nodes broadcast data with the assistance of power control in Figure 7.1(a), encoded packets are generated in each node, shown in Figure 7.1(b). The generated encoded packets in ECPC can guarantee a successful data recovery under the certain failure probability ϕ, e.g. 10% here. For continuous sensor data stream, ECPC only needs to repeat distributed erasure coding after network initialization. We present the

in-depth description of ECPC designs and analysis in the following sections.

6.2 ECPC Algorithm Design and Analysis

ECPC is a distributed approach to preserve data persistence in disruptive networks. In this section, we describe the distributed ECPC algorithm which contains two major procedures: (1) *Randomized Power Control*; (2) *Distributed Erasure Coding*. The theoretical analysis is given to validate its correctness and effectiveness.

6.2.1 Randomized Power Control

In disruptive network, only partial nodes are present in the network when the data retrieval is conducted. Thus, the redundancy of encoded packet must be introduced to compensate the data lost in failure nodes. This redundancy is to ensure that the sufficient number of packets are available to decoder whenever the retrieval is conducted.

In any retrieval time, the expected number of alive nodes equals to $n(1-\phi)$. Thus, the available encoded packets is: $n(1-\phi)\cdot\Gamma$, where Γ is the amount of encoded packets per node. In the other end, according to Raptor Codes and randomness analysis, the decoder requires $\frac{1}{\Lambda}(1+\epsilon)\cdot n$ packets to successfully recover the original n data, shown in *Lemma 2* of Section 6.2.3. To determine the adequate redundancy, the necessary encoded packets per node is $\Gamma_{min} = \frac{1+\epsilon}{(1-\phi)\Lambda}$, where $\Lambda = (0.48 + 0.06\ln\frac{\delta}{6})$ and $0 < \delta < 1$. δ is the probability of decoding failure in Raptor Codes, which is an adjustable parameter in Raptor Codes design. Notice that Γ is configurable as long as it is larger than Γ_{min}. Larger Γ means higher redundancy and higher data recovery ratio.

Once determining the redundancy per node, a novel randomized transmission power control is applied. This randomized power control is conducted Γ times to generate the desired encoded packets in each node. The purpose of this power control is to make the node degree distribution conform to the expected code degree distribution $\Omega(Y)$. We transform the design paradigm of drawing a code degree from a known degree distribution for encoding. Instead, code degree is auto-determined by the actual data packets received from neighbors,

i.e. actual node degree. The node degree is determined by the transmission power level of neighbor nodes in the vicinity.

Towards this goal, we derive a posterior probability distribution $f_P(P_i)$, taking the code degree distribution $\Omega(Y)$ as the known observation. Then randomized power control can be achieved by selecting on power level from derived power distribution once per iteration.

The key approach to deriving the power distribution is based on Bayes' Theorem [73]. Define Y as the event for code degree selection. Since we consider a random graph, the node distribution conforms to the Poisson distribution. Thus, according to Poisson Distribution, given an event of transmission power $\{P = P_i\}$, the likelihood function of Y is denoted as:

$$f_Y(y|P = P_i) = \frac{\lambda^y e^{-\lambda}}{y!} = \frac{(\eta\pi R^2(P_i))^y e^{-\eta\pi R^2(P_i)}}{y!} \tag{6.1}$$

where e is the Euler's number, $\lambda = \eta\pi R^2(P_i)$ is the expected node degree. When each node is assigned with the same transmission power P_i, then the probability distribution of neighbor number is defined by this likelihood function. Since each node holds the same transmission power, resulting in symmetric link conditions, the number of different nodes it can cover equals to the number of its neighbors. Thus, the node degree can be described by λ in Equation (6.1).

Next we can compute the posterior probability distribution of transmission power:

$$\begin{aligned} f_P(P_i|Y = y) &= \frac{f_{P,Y}(P_i, y)}{\int_{-\infty}^{+\infty} f_Y(y|P = \xi) f_P(\xi) d\xi} \\ &= \frac{f_Y(y|P = P_i) f_P(P_i)}{f_Y(y)} \\ &= \frac{f_Y(y|P = P_i) f_P(P_i)}{\Omega(y) \cdot (1/\varphi)} \end{aligned} \tag{6.2}$$

where $f_P(P_i)$ and $f_Y(y)$ are the marginal probability density functions of P and Y respec-

tively. Without any prior knowledge about the event P, we provisionally assume the prior distribution of P, $f_P(P_i)$, is uniformly distributed over the interval $[P_{min}, P_{max}]$. And $f_Y(y)$ is expected to be $\Omega(y) \cdot (1/\varphi)$, so that encoding any φ $(0 < \varphi < 1)$ portion of received data can make the code degree distribution conform to $\Omega(\cdot)$. Meanwhile, $f_Y(y|P = P_i)$ can be calculated by Equation (6.1) accordingly. As we already characterize the distribution of Y for each event $\{P = P_i\}$ in Equation (6.1), the probability distribution of event P can be computed by Equation (6.2), given the probability distribution of event $\{Y = y\}$. The final power distribution is derived as: $f_P(P_i) = \sum_y f_P(P_i|Y = y)$. Note that the normalization on $f_P(P_i)$ is conducted thereafter to make the sum of probability equal to 1.

Algorithm 7 Randomized Power Control of node u

Input: *Code degree distribution* $\Omega(Y)$, Γ, n

1: **while** $P_{min} \leq i \leq P_{max}$ **do**
2: **if** $\mathcal{N}_u^i = 0$ **then**
3: Broadcast a beacon message with P_i
4: Start a timer with ΔT fire interval
5: **end if**
6: Upon reception of beacon message
7: Reply ACK with the same power level P_i
8: Upon reception of a ACK message
9: Increase neighbor counter $\mathcal{N}_u^i \leftarrow \mathcal{N}_u^i + 1$
10: **if** ΔT fire **then**
11: Get the neighbor size under power P_i: $\mathcal{N}_u^i$
12: **end if**
13: $i \leftarrow i + 1$
14: **end while**
15: Compute the $f_P(P_i|Y = y)$ according to Equation (6.2)
16: Derive power distribution: $f_P(P_i) = \sum_y f_P(P_i|Y = y)$

Next we describe the power control algorithm of ECPC (Algorithm 7) to derive the above power distribution, and conduct the randomized control accordingly. In Algorithm 7, transmission power level is controlled by a power distribution $f_P(P_i)$. Firstly, line 1 computes the lowest power level P_{min} which guarantee that the actual node degree is larger than the expected code degree in each node. Secondly, procedure from line 2-15 collects the neighbor size information for each power level per node. Thirdly, based on Equation

(6.2), the $f_P(P_i|Y = y)$ is computed, and hence $f_P(P_i)$ is derived. It is worthy to mention again that, when network condition does not change, the power distribution only needs to be calculated once at the initialization. Normally, only distributed erasure coding is performed at each sensing round, as presented in next section.

6.2.2 Distributed Erasure Coding

In this section, we present the design of distributed erasure coding in Algorithm 8 and its analysis. Since Raptor Codes is the first known class of fountain codes with linear time encoding and decoding, our design uses Raptor Codes as the encoding and decoding approach. Raptor codes relax the condition that all the input symbols required to be recovered by introducing extra intermediate encoded parity-check data [18]. Therefore, the LT codes operated in the second phase have a decoding complexity of $O(n)$.

Raptor Codes encoding contains two-phase operation, we first discuss the construction of intermediate parity check symbols in the pre-coding phase. We denote this pre-code as $\mathcal{C}_m(\lambda, \rho)$, with the coding rate as $(r = 1 - \frac{\lambda}{\rho})$. LDPC (Low-Density Parity-Check) codes are proven to be a competitive codes for linear block coding purpose. LDPC codes exhibit a linear encoding complexity in many cases. A $\mathcal{C}_m(3, 6)$ regular LDPC, with input symbol length of k, consumes the actual number of operations no more than $0.017^2k^2 + O(k)$. Thus even large block lengths exhibit quite manageable encoding complexity [74]. Therefore, our design specifically implements $\mathcal{C}_m(3, 6)$ regular LDPC codes for pre-coding in a distributed fashion.

The key to the construction of LDPC codes is generating an appropriate *parity-check* matrix H, such that $Hx^T = 0$. In Figure 6.3, we show an example of *parity-check* martix H for $\mathcal{C}_m(3, 6)$, with 10 input symbols and 5 output check nodes. In $\{0, 1\}$-matrix $\mathbf{H}$, a nonzero entry at row i and column j indicate a connection between the jth symbol data and ith parity-check data. In $\mathcal{C}_m(3, 6)$, the sum of component 1 in each column should equal to 3, and summation of component 1 in each row should be 6. The *parity-check* martix H can be constructed in a distributed manner illustrated in Pre-Coding Phase of Algorithm

8. By using $\mathcal{C}_m(3,6)$, we generate $n/2$ *parity-check* nodes, resulting $m = 1.5n$. Therefore, every node elects itself as parity-check node with $\frac{m-n}{n} = 50\%$ probability. And each node broadcasts a message indicating its parity-check node status, waiting for the contribution messages.

$$
H = \begin{array}{c}
\begin{array}{cccccccccc} V1 & V2 & V3 & V4 & V5 & V6 & V7 & V8 & V9 & V10 \end{array} \\
\left[\begin{array}{cccccccccc}
1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1
\end{array}\right]
\begin{array}{c} C1 \\ C2 \\ C3 \\ C4 \\ C5 \end{array}
\end{array}
$$

Figure 6.3 An example for a parity-check matrix $\mathbf{H}$.

The second phase of Raptor Codes is to directly apply LT codes with the m input symbols from the first phase. As customary, our Raptor codes scheme is specified by parameter tuple $(n, \mathcal{C}_m(\lambda,\rho), \Omega(x))$.

In Algorithm 8, we propose a distributed erasure coding algorithm. Procedure *pre-coding* phase constructs a regular (λ,ρ) LDPC codes. In particular, we use $(3,6)$ LDPC codes. In line 6, each node u contributes to λ of parity-check nodes in its vicinity, whose parity-check matrix H is not satisfied. Thus, code constructions from line 2-9 can satisfy the matrix H's requirement. Note that the (λ,ρ) can be altered to have different performances of LDPC [16].

Each node locally draws a transmission power level P_i from $f_P(P_i)$, and broadcast its symbol with power P_i. It can be observed that if the $\mathcal{N}_u$ equals to $n-1$, the global randomness in original RSD is preserved. As $\mathcal{N}_u$ decreases, we obtain a pseudo-global randomness in the neighbor symbol selection, because node u no longer has full access to other nodes in the network, like the case in complete graph. The analytical proof in Section 6.2.3 shows that decoding cost for successful recovery is bounded, even when node u selects its input symbols from its current neighbor list only.

Algorithm 8 Distributed Erasure Coding of node u

Input: Γ and $f_P(P_i)$

procedure: *Pre-Coding Phase:* (λ, ρ) *regular LDPC*

Elect as parity-check node with probability λ/ρ
if Node u is parity-check node **then**
  Constraint status $\mathcal{C}(u) = 0$
  while $\mathcal{C}(u) < \rho$ **do**
    Broadcast its $\mathcal{C}(u)$
    Send message to its parity-check neighbors, whose $\mathcal{C}(v) < \rho$
    Upon reception of contribution message
      Update $\mathcal{C}(u) \leftarrow \mathcal{C}(u) + 1$
  end while
  Encode the contribution message to parity-check packet
end if

procedure: *LT-Coding Phase*

while $\Gamma > 0$ **do**
  Randomly draw transmission power P_i from $f_P(P_i)$
  Broadcast its own sensor data under power P_i
  if Node u is parity-check node **then**
    Broadcast parity-check packet
  end if
  Upon receiving packet S_v from neighbor v
    Store S_v into encoding pool
  $\Gamma \leftarrow \Gamma - 1$
end while
Randomly select $\binom{\mathcal{N}_u}{\lfloor \varphi \cdot \mathcal{N}_u \rfloor}$ symbols
Encode packet $E_u = S_1 \bigoplus S_2 \bigoplus ... \bigoplus S_{\lfloor \varphi \cdot \mathcal{N}_u \rfloor}$

6.2.3 Analysis on Pseudo Randomness

In this subsection, we perform a theoretical analysis of the impact of pseudo randomness on the decoding performance.

Inspired by the proof in [75], we prove that during recovery process, the occurrence of neighbor symbols appear to be independent from each other. We model the recovery set S by randomly selecting input symbols out of k, and put them into set S with probability $p = \frac{n-L}{n}$. L is the set of not yet recovered input symbol: $n = |L| + |S|$. Let X_i be the indicator random variable of the event that a input symbol belongs to set S, so X_i is an independent variable. Then, we have the following lemma.

Lemma 29 *Given an encoded packet of degree d covers an input symbol, at least a $(1-\delta)$ fraction of sets S ($0 < \delta < 1$) satisfy: for all $m \notin S$, the probability that the encoded packet covers m is at least $(1 - 4\beta^2(16 - 2\ln(\delta/6)))\frac{1}{L}$, where $0 < \beta < 0.1$.*

Proof 30 *We first define f_m as the probability that symbol m is covered when some packet of degree d has been released.*

$$\begin{aligned} f_m &= \frac{\sum_{i \in \mathcal{N}_m} X_1 ... X_{m-1}(1 - X_m)X_{m+1}...X_i...X_d}{\sum_{i \in G} X_1 ... X_i ... X_d} \\ &\doteq \frac{A}{B} \end{aligned}$$

Conditioned on $m \notin S$, the expected number of A can be computed as $\mathbb{E}[A] = p^{d-1}d\pi\eta P_E^2$, where P_E is the expected transmission power derived from $f_P(P_i)$. Since m is covered, its $d-1$ neighbors should already be in the set S. Thus, we have p^{d-1} term. The item $d\pi\eta P_E^2$ is the number indicating the total combinations of degree d encoded packet holding symbol m in the network. For denominator, we have $\mathbb{E}[B] = p^{d-1}(1-p)dn\eta P_E^2$, and $\mathbb{E}[A]/\mathbb{E}[B] = \frac{1}{n(1-p)} = 1/L$. We derive a lower bound on f_m by proving the A and B are close to their expected values respectively.

We first apply Janson Inequality [76] to bound the A from below. In Janson Inequality, we define $\Delta = \sum_{A_i \sim A_j} \mathbb{P}[A_i \bigwedge A_j]$, *where* $A_i \sim A_j$ *means that the intersection of these two set are not empty. Let* $A = \sum_i A_i$, *for any* $0 < \sigma < 1$, $\mathbb{P}[A \leq (1-\sigma)\mathbb{E}[A]] \leq \exp(-\frac{\sigma^2 \mathbb{E}[A]}{2+\Delta/\mathbb{E}[A]})$. *Recall that* $\mathbb{E}[A] = p^{d-1} d\pi\eta P_E^2$. *Pairs of* $A_i \sim A_j$ *of degree d have d options to hold symbol m respectively, the total number of this is calculated as* d^2. *Besides m symbol, dependent pairs should have at least one symbol in common, with each of them having d choices in positions of encoded packets. It renders another* d^2 *possible combinations. Since dependent nodes are with the vicinity of node producing m symbol, the maximum number of choices common symbols is upper bounded by* $\pi\eta P_E^2$. *Moreover, if* A_i *and* A_j *are dependent,* $\mathbb{P}[A_i \bigwedge A_j] \leq p^{d-1}$, *where two sets share all the encoding neighbors. Thus,* $\Delta \leq p^{d-1} d^4 \pi\eta P_E^2$. *In the following deduction, we let* $\sigma = \frac{\alpha d^2 \ln n}{(n-1)(1-p)}$, *where* $\alpha = \frac{16 - 2\ln(\frac{\delta}{6})}{\ln n}$ *[75]. Since* $\mathbb{E}[d] < n/R = \sqrt{n}/c \ln(n/\delta) < \beta\sqrt{n}$ *according to LT codes [17], then* $\mathbb{E}[d^2] < \beta^2 n$. *Therefore,* $\sigma < (16 - 2\ln(\delta/6))\beta^2$.

Next, based on Janson bound we can obtain:

$$\begin{aligned}
\mathbb{P}[A \leq (1-\sigma)\mathbb{E}[A]] &\leq \exp(-\frac{\sigma^2 \mathbb{E}[A]}{2 + \frac{\Delta}{\mathbb{E}[A]}}) \\
&\leq \exp(-\frac{\frac{\alpha d^2 \ln n}{(n-1)(1-p)} d\pi\eta P_E^2}{2 + d^3}) \\
&\leq \exp(-\frac{\alpha d^3 \cdot \ln n \cdot \frac{\pi\eta P_E^2}{n-1}}{3d^3}) \\
&= n^{-\frac{\alpha}{3}} \qquad (6.3)
\end{aligned}$$

Thus, $\mathbb{P}[A \geq (1-\sigma)\mathbb{E}[A]]$ *is at least* $(1 - n^{-\frac{\alpha}{3}})$. *Then we bound denominator B with no conditioning on* X_m, *using Chernoff bound as described in [77]. We can have:* $\mathbb{P}[B \leq (1+3\sigma)\mathbb{E}[B]] \geq 1 - n^{3-\alpha}$.

Now we can get the lower bound for f_m: $f_m \geq (1-4\sigma)\frac{1}{L} \geq (1 - 4\beta^2(16 - 2\ln(\delta/6)))\frac{1}{L}$ *with probability at least* $(1 - n^{-\frac{\alpha}{3}}) \cdot (1 - n^{3-\alpha}) \geq (1 - 3n^{8-\frac{\alpha}{3}}) \geq (1 - \frac{\delta}{3}) > (1 - \delta)$. *Under the system parameter* $\delta = 0.05$, *the range of* β *can be determined. The upper bound of* β *is*

0.1, *which makes* $\Lambda > 0$. *When* n *increases further, the* β *can be lower, as long as making* $\mathbb{E}[d^2] < \beta^2 n$ *hold. The proof is complete.*

Theorem 31 *Given the network of* n *sensor nodes, power control assisted encoding process (ECPC) can recover* n *input symbols with probability* $(1-\delta)$ *even with* $1-(1+\epsilon)/(\Gamma \cdot (0.48 + 0.06 \ln \frac{\delta}{6}))$ *percent node failures.*

Proof 32 *During decoding period, set* S *is the recovery set containing the symbols that have been recovered, and set* L *holds the uncovered symbols. To prove the randomness in neighbor symbol selection, it is critical to prove that the distribution of the set* S *of already recovered input symbols remains uniformly distributed throughout the execution of the recovery process. In Lemma 1, it has been proved that with probability at least* $(1-\delta)$*, we can lower bound the release probability of every uncovered symbol of set* L *with* $\Lambda\frac{1}{L}$*. That is to say, it needs* $\frac{1}{\Lambda}$ *encoded packet to cover each uncovered symbol with probability* $\frac{1}{L}$ *for at least* $(1-\delta)$ *percent case. Since* $\beta < 0.1$*, it is the worst case of* Λ *when* $\beta = 0.09$*. Thus,* $\Lambda = (0.48 + 0.06 \ln \frac{\delta}{6})$.

6.3 Protocol Implementation

ECPC protocol is implemented in TinyOS [70], which supports light-weight embedded system design. In Figure 6.4, we show ECPC protocol implementation in the shaded boxes, which are core parts of the entire system of distributed data storage. The supporting components in system include data sensing, flash storage, MAC protocol for local broadcasting and radio communication drivers.

As illustrated, the ECPC protocol consists of two shaded components: *ECPC Encoding* and *ECPC Power Control*. In *ECPC Power Control*, a posterior distribution of transmission is calculated with two input information, which are size of neighbor nodes and appropriate degree distribution $\Omega(*)$. The component "Tx Power Distribution" is responsible for adjusting Tx power in MAC layer for one-hop message broadcasting, through a command control interface. We select the degree distribution parameters as follows: $\delta = 0.1$, $\epsilon = 0.05$, and $\varphi = 0.5$ for evaluating transmission power distribution $f_P(P_i)$ in Equation 6.2. In *ECPC*

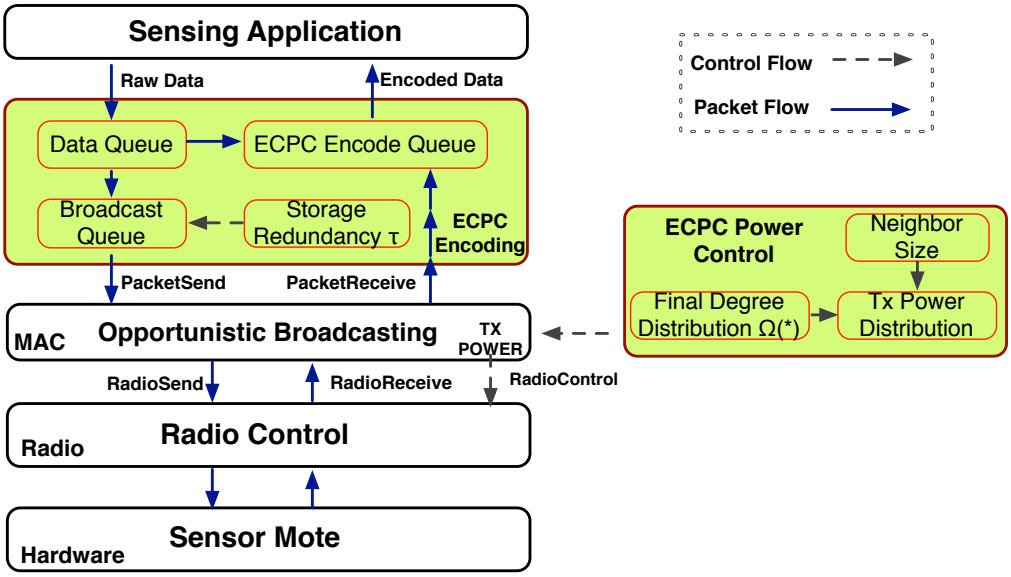


Figure 6.4 ECPC Protocol Implementation

Encoding, storage redundancy τ is first determined by knowing the failure probability Φ and deployed network size N. The obtained τ is used to rein the broadcasting cycle of raw data. Data queue is implemented to buffer the sensing data for two purposes: (1) encoding sensing data with data from other nodes; (2) inserting into broadcast queue for one-hop broadcasting. ECPC encoding utilizes network erasure coding to encode data, including pre-code stage (LDPC) and LT-code stage (LT Codes). We select system parameter $\lambda = 3$ and $\rho = 6$ for pre-code LDPC.

As a full protocol implementation, ECPC is light weighted in two aspects. First, the data offloading for distributed storage solely counts on one-hop broadcasting, without obligation of maintaining packet forwarding probability matrix in every node for random walks. Second, final packet degree distribution is accomplished by broadcasting with ECPC transmission power control, reducing the complexity in tracing packet among multiple random walks.

6.4 Performance Evaluation

In this section, we present the evaluation of the proposed ECPC algorithm, comparing to the existing approaches for distributed data storage: Exact Decentralized Fountain Codes

(EDFC) in [39] and Raptor-Codes based Distributed Storage (RCDS) in [41]. EDFC disseminates data through multiple random walks to ensure the data received per node is no less than code degree selected. The transition matrix is constructed by Metropolis Algorithm with stop probability proportional to code degree selected. RCDS initiates only one random walk for each data item, which has length of $O(nlogn)$ to ensure this random walk visit each node at least once. Every node accept the random walk packet with a specific probability and encode by Raptor Codes. The ECPC and compared approaches are implemented in TOSSIM simulator [69].

We model the network as a Random Geometric Graph [72], which is deployed in a area $\mathcal{A} = [D, D]^2$, where D is the length for each of dimension of deployed region. In the simulation, we fix the area as $\mathcal{A} = [20, 20]^2$, but vary the network size n. Since the network density can be computed as: $\eta = n/D^2$, different network sizes denotes different network density as well. An independent failure probability is associated with each node. The adjustable radio power range is given as $[95, 127]$, which maps to the radio transmission range $[1, 8]$, with 32 levels in total. To alleviate the uncertainty of single experiment, we average each of our results from 50 separate experiments.

6.4.1 Communication Overhead

We examine three metrics under different network sizes: *Message Cost, Energy Cost, and Termination Time*. In Figure 6.5, we compare the total message cost of our ECPC algorithm with EDFC and RCDS. In general, the total message cost of ECPC is much lower than other two approaches. And the message cost difference increases faster as the network size scales up. This is because that the dissemination in EDFC and RCDS rely on the random walks, which is at least $O(n^2 \cdot polylogn)$, while the dissemination of ECPC only consumes $O(1)$ message per node. In particular, the message cost of ECPC stays under $1,000$ under the network size of 500. RCDS requires $2,000,000$ messages in total to achieve the expected code degree distribution across the network. Approximately $1,200,000$ messages is consumed in EDFC.

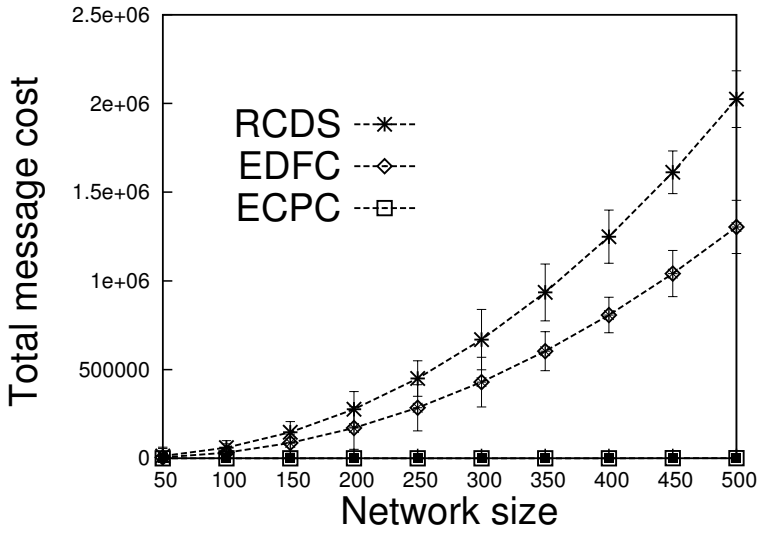


Figure 6.5 Communication overhead: total message cost

Since the packet transmission power is dynamically adjusted in ECPC, energy cost for every message broadcasting indeed vary dramatically. Thus, reduction of message cost does not guarantee a better energy efficiency. We consider the total energy consumption required for protocol, avoiding the arguable concerns about the energy efficiency of proposed ECPC. In Figure 6.6, the measurement of total energy ratio between *RCDS* and *ECPC*, *EDFC* and *ECPC* are presented respectively. For RCDS, the energy ratio to ECPC could rise from about 20 ($N = 100$) to 100 ($N = 500$). Compared with RCDS, EDFC decreases the number of random walks for each raw data by reducing the exact condition to approximated conditions. Hence, the maximum energy ratio between EDFC and ECPC is about 60 under network size 500. The reason behind this significant energy saving is that ECPC only adopts single-hop packet broadcasting to ensure the final degree distribution is satisfied among encoded and stored packets. Though transmission power may be increased to cover more neighbors in one broadcasting attempt, the expected Tx power value throughout multi-round encoding process is closer to $\bar{P}$.

In terms of time for data dissemination, ECPC method terminates at a short time period, e.g. less than 10 time units under varying network sizes, shown in Figure 6.7. In EDFC and RCDS, each source node can initiate its own random walk at the same time, as

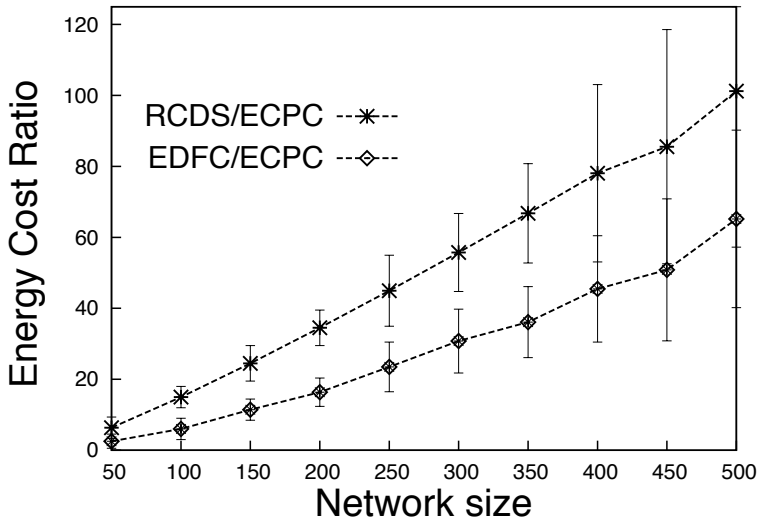


Figure 6.6 Total energy consumption in distributed data storage schemes.

far as there is no media access conflict. Their termination of encoding did cost a considerable time, with 200 time units for EDFC and 400 time units for RCDS respectively under the network size of 500.

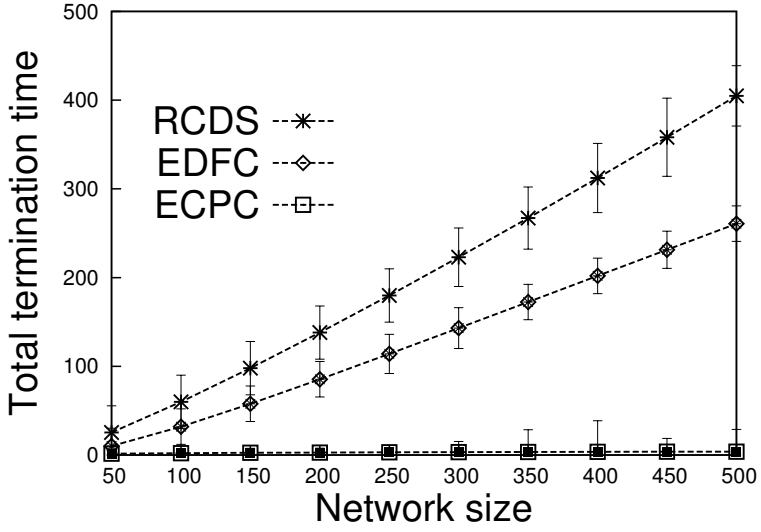


Figure 6.7 The communication overhead: total termination time.

6.4.2 Data Recovery Ratio

Illustrated in Figure 6.8, we study the decoding performance in terms of data recovery ratio. Data recovery ratio denotes the percentage of original data recovered after decoding. The node has a failure probability of 10%. First, we observe that the recover ratio of ECPC has 40% at the network size 50, but, increase to over 90% as network size increases beyond the 400. In experiment, ECPC select a β of 0.07 from range $(0, 0.1)$ to maintain the randomness properties. The decoding performance of EDFC and RCDS also experience an increasing trend as network size grows. The reasons for their poor performance is that nodes can fail when the random walk is still going. It may result in a stop condition for the random walk on that failure node. Thus, the resultant degree distribution is not the best match of expected code degree distribution. In particular, the data recovery ratio is only 67% and 62% for EDFC and RCDS respectively.

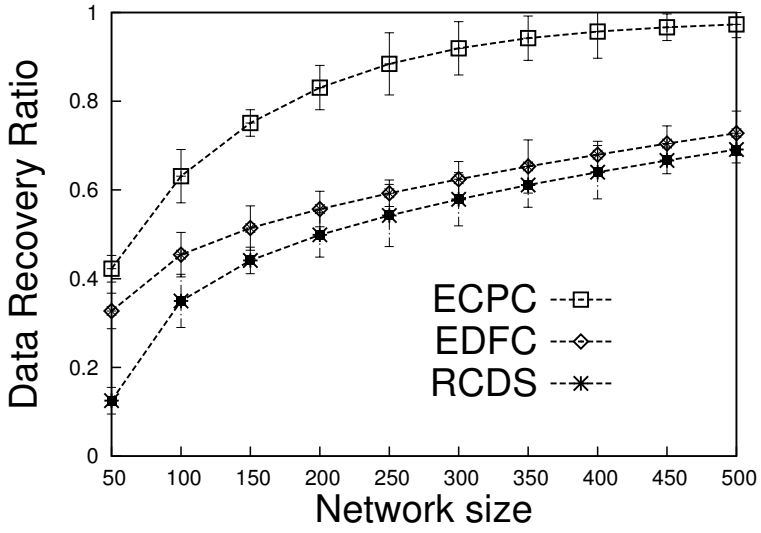


Figure 6.8 The decoding performance under varying network sizes: Max data recovery ratio for different network sizes

We examine the data recovery ratio for 500 nodes along the axis of experiment time elapsed in Figure 6.9. We keep node failure probability as 10%. Since we repeat the experiments 50 times with node failure occur randomly each time, the overall failure occurrence time is also uniformly distributed. Since EDFC and RCDS need 300 to 400 time units to

perform the random walk based data dissemination, the recovery ratio is 0 during the early time period of data dissemination for both EDFC and RCDS. On the other hand, ECPC terminates in a short time period, making data recovery possible even in a early stage, like at the time of 100 time units. The peak recovery ratio of EDFC can only reach 67% is due to the node failure occur during the data dissemination. As the time elapses, the recovery ratio of three approaches decrease. It is because that more nodes fail as time elapses. However, the decoding performance of ECPC degrades much less than other two approaches.

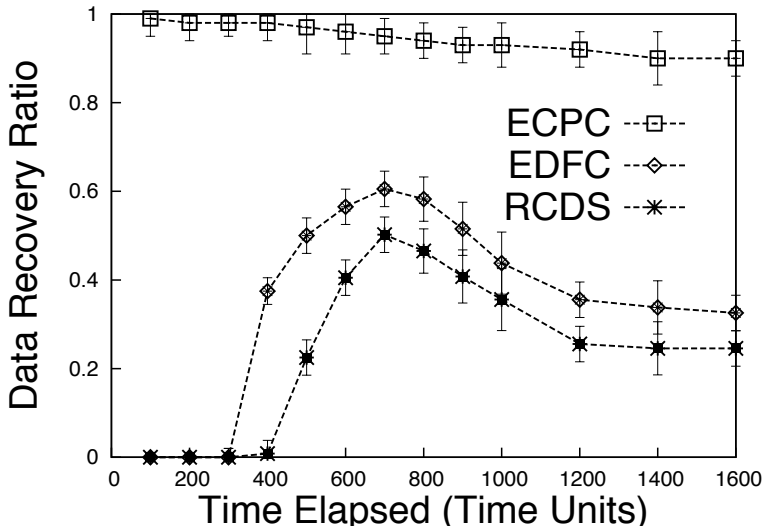


Figure 6.9 The decoding performance under varying network sizes: sequential snapshots of data recovery ratio as time elapses.

6.4.3 Data Recovery under Disruptive Networks

In this subsection, the impact of varying failure probabilities ϕ on data recovery ratio is shown in Figure 6.10. The network size is 1000 nodes. In Figure 6.10, it is shown that the three approaches have higher data recovery ratio in case of lower node failure probability. For instance, in the case of 10% failure probability, ECPC's recovery ratio is close to 100%, and EDFC and RCDS have 90% and 82% recovery ratio respectively. However, as the failure probability increases, the recovery ratio in EDFC and RCDS decreases significantly. At the network of 80% node failure probability, ECPC can still preserve 80% data on average, which

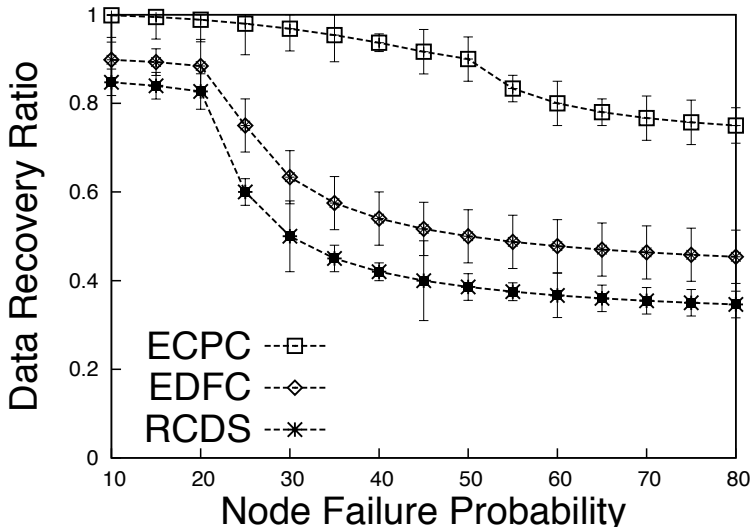


Figure 6.10 Recovery ratio under varying failure probabilities

is 160% as much as the decoding ratio of EDFC, and 200% as much as the that of RCDS. It is due to two reasons. First, ECPC has a fast termination time. It can reduce the chance of experiencing disruptive nodes, which may halt the data dissemination process. Second, redundant packets, determined in Section 6.2.1, account for the failure node, by encoding extra packets in each node.

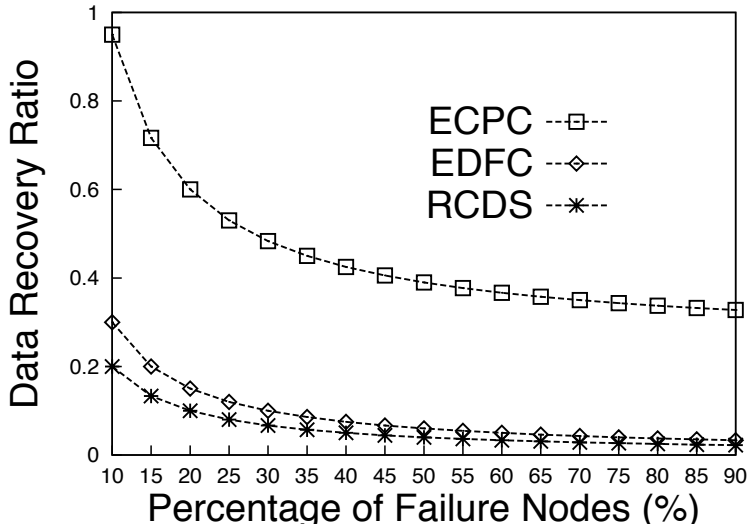


Figure 6.11 The decoding performance with varying percentage of failure nodes at early encoding stages.

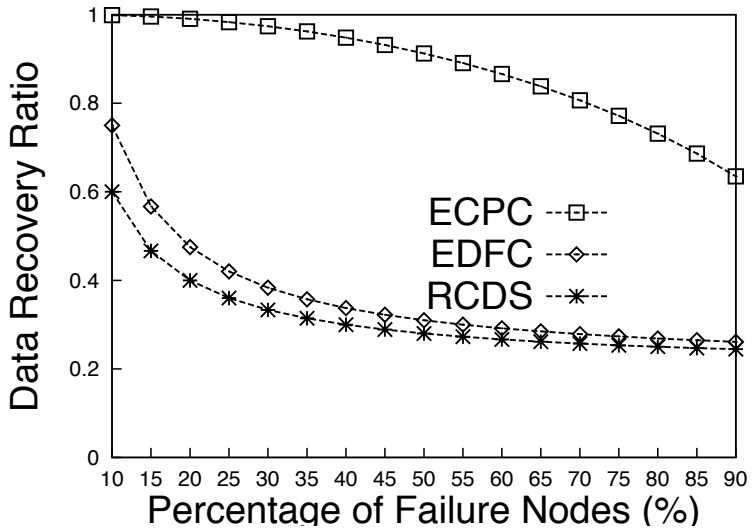


Figure 6.12 The decoding performance with varying percentage of failure nodes at middle encoding stages.

We further investigate the impact of disruptive nodes in different stages of data dissemination and encoding. According to the experiment in Figure 6.9, we categorize the stages into two stages: early stage ($0-100$ time units) and middle stage (100-600 time units), shown in Figure 6.11 and Figure 6.12 respectively. In Figure 6.11, early node failures compromise the decoding ratio of ECPC most, making recovery ratio 40% in case of 90% node failures. In the other hand, node failure during middle stage has less impacts on ECPC. Under 40% failure nodes, ECPC can recover over 95% data. However, the best recovery ratio of EDFC and RCDS are less than 30% and 70% under (a) and (b) respectively.

6.4.4 Evaluation of Long-term Stability

To preserve the persistence of real-time data stream in a disruptive network, distributed erasure coding with randomized power control run repeatedly to offload, encode and store data. ECPC protocol re-evaluates a new posterior distribution of transmission power only when the number of neighbor nodes change significantly.

The long term stability of ECPC is to evaluate the decoding performance of data recovery only based on initial transmission power distribution. Long-term stability is shown in Figure 6.13, in the case that power control algorithm is only conducted once at the initial-

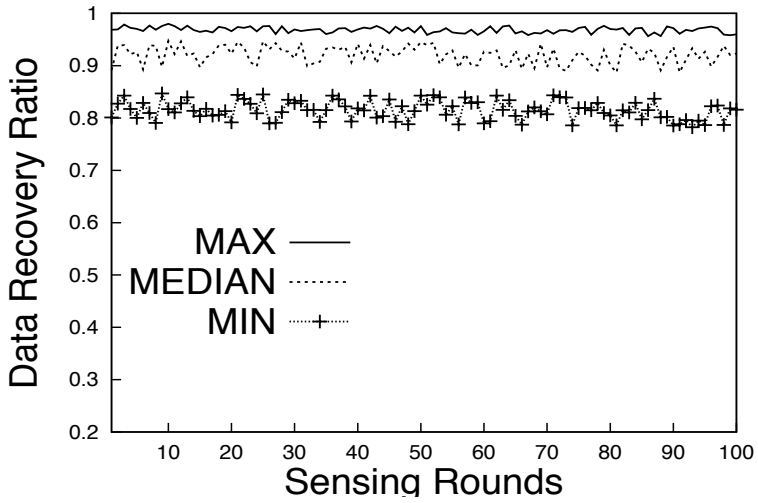


Figure 6.13 Recovery ratio under different sensor periods

ization. In the simulation setup, we set the disruption probability of each node as 10%. The MAX, MEDIAN and MIN curves illustrate the maximum, median and minimum recovery ratio during the long-term stability evaluations. The uncertainty of disruptive nodes during sensing period can compromise decoding performance by 10% at most. However, even under inaccurate estimation, ECPC decoding is stable, which recovers more than 90% data by decoding packets on average (MEDIAN curve).

CHAPTER 7

INTEGRATED SOLUTION FOR DATA COMMUNICATION AND STORAGE IN ENERGY-SYNCHRONIZED SENSOR NETWORKS (RAVINE STREAM)

In a large-scale sensor network deployment, one of major reasons contributing to intermittent connections and disruptive communication is sensor nodes failure. Note that the reasons for node failure have a quite broad range, which can be either power depletion, unable to route packet to destination or overflowed receiving and transmitting buffer. Each node failure influences the data collection flow routed through it. In the worst case, sink node failure impairs the entire network traffic. To preserve data persistence, data stream has to be rerouted to other paths or redistributed to other storage nodes if none active routes are found. Existing algorithms from the literature studied this data preserving as an network flow optimization problem, like [57] maximizes the minimum residue energy in the network for the next data redistribution to arise; [55] combines minimizing data redistribution and retrieval cost into a single problem.

Though optimal solutions are devised under either energy or storage space constraints, most of the previous works ignore the fact that node failures could happen in the course of data rerouting or redistribution. The determined approach is not capable of scheduling data redistribution without the global knowledge of which part of nodes could fail in the middle. Moreover, the data redistribution is conducted in the network of heterogeneous node failure probabilities. This fact implies that data should move towards more reliable nodes with sufficient residue energy and storage space, such that higher data delivery rate can be achieved.

In this paper, we propose *Ravine Streams* to preserve data stream persistence in disruptive sensor networks. Source data is initiated and delivered in its encode form by network

erasure codes *OnCode* [78]. For data preservation, receiving nodes distributively make the acceptance decision based on local failure probability and storage space. Note that here nodal failure probability has taken the residual energy into account and dynamically update across the process. Adaptive transmission power control in *RS* ensures that data acceptance probability by neighbor nodes is expected at 1 for each broadcasting under minimum transmission power. With data packet recoding, the dispensable data content redundancy is constrained for more energy efficiency. Moreover, we show that the delivery performance of proposed *RS* is bounded by *Optimal* solution: $\eta = \frac{OPT \cdot \ln(\frac{B}{\delta})}{2\lambda}[1 + \omega(1) \cdot \frac{1}{B} + 2\omega(2) \cdot \frac{B-2}{\binom{B}{2}}]$.

In summary, this work has three contributions: (1) RS leverages probabilistic transmission power control and network erasure recoding to preserve data throughout disruptive network connection. Additionally, the data redundancy is largely reduced, increasing algorithm energy efficiency. (2) RS makes distributed probabilistic decision that directs redistribution data to more reliable storage nodes, without global knowledge of density and node failure probabilities, smoothing data collection traffic under disruptive connections. (3) Our analysis shows that the proposed RS algorithm has bounded performance compared with optimal solution.

7.1 Ravine Stream Algorithms and Analysis

Ravine Streams is an integrated set of distributed algorithms that protect data collection stream from disruptive network nodes and connections. The data persistence preservation of *RS* is built upon the opportunistic in-network coding and delivery (OnCode) [78]. Data packets are initially encoded and transmit by *OnCode*. When local storage space vanishes, *RS* redistributes excessive received data packets to more reliable nodes to preserve data persistence. RS considers both heterogeneous failure probabilities (due to energy depletion or other factors) and storage constraints, and addresses the problem of disruptive connection during data redistribution. RS leverages network erasure coding in data redistribution, and determines the rebroadcasting probability and transmission power level according to nodal utilities. It is worth to mention that RS algorithms are probabilistic, adapting to the dynamic

network conditions with minimum execution overhead.

We describe the properties of an ideal data recode and redistribution algorithm, which is referred to as *OPT*. As the optimal benchmark solution, *OPT* presents the following properties that our algorithm design concerns. First, *OPT* does know how much data can be accepted and how much data needed to be redistributed if the node happens to be disruptive at a certain time. Here, we define a node is disruptive as it can not forward any data towards sink, and data stream is disconnected hereafter. In other words, *OPT* determines the *data acceptance* in an optimal way so that the storage use are balanced across the network. Second, for a data redistribution in a time range ahead, *OPT* can always find a determined path from disruptive node to destination storage node with stable communication link and ensured data delivery. And only one copy of source data packet needs to be maintained inside network. In practice, it is unlikely to predict a stable communication link across multi-hop connection in the disruptive network conditions, which means the data redistribution is not guaranteed to be successful every time. Thus, multiple copies of data packet are necessary to reduce the risk of data loss. Under this condition, data redundancy needs to be reduced between multiple survived data copies. Third, *OPT* can observe if the disruptive node is trapped in bad region or not, so as to adjust the appropriate radio transmission power to assist the communication. In fact, without perfect knowledge ahead of time, it consumes considerable overhead to search different levels of transmission powers until a proper power is found.

RS adopts two integrated mechanisms to tackle the unideal situations: *Determine Data Acceptance*, *Probabilistic Data Redistribution*. RS is able to preserve data persistence in heterogeneous situations subject to node failure probability and storage constraints. The mathematical symbol and notations are described in Table 7.1.

Table 7.1 Notation in Algorithm

Notation	Explanation
N_i^m	Neighbor set of node i under power level m
U_i^m	Nodal utility under TX power m
δ_i	Failure probability of node i
C_i	Free storage space on node i
C_{max}	Maximum storage space
β	System parameter for data acceptance
α	Replacement factor
q_i	Probability of accepting incoming packet in node i
p_m	Selection probability for power level m
τ	Exponential index of power adjustment

7.1.1 Determine Data Acceptance

Due to the opportunistic nature of encoded network traffic, it is unlikely to predict the exact amount of incoming packet in a certain time window. Therefore, when disruption occurs, there is hardly a determined way to calculate the exact amount of data needed to be redistributed. The best approach is to make this decision based on the dynamic available storage space in each node. RS adopts the reactive methods to discern available storage in node i, defined as C_i. We first look at an example of a data stream from $i \rightarrow j \rightarrow k$, where node k becomes disconnected, making node j not able to forward any incoming traffic. Node j stores data packets unsent in its own buffer. To preserve data persistence, node j has to redistribute the excessive data to other nodes for storage before its space is used up.

If data packets are redistributed only when storage space is full, the network will experience a sudden and unexpected traffic burst. To avoid this dramatic traffic change, RS adopts probabilistic method to pre-estimate the probability q_j for each incoming packet. The probability q_j indicates the probability with which node j accept the received packet. With this smoothing technique, network traffic can remain relatively stable even under unforeseen

disruptive conditions.

The probability q_j is determined distributively, based on local ratio of nodal available storage space C_j and maximum storage space C_{max}. With less available storage ratio, incoming packet will be less likely to be accepted. Moreover, each node i has a certain failure probability δ_j due to either energy depletion or other external factors. Therefore, final data acceptance probability is estimated against factors $\frac{C_j}{C_{max}}$ and δ_j. We combine these two factors by *multiplicative* operation.

$$q_j = \beta \cdot U_j = \beta \cdot (\frac{\delta_j \cdot C_j}{C_{max}}) \tag{7.1}$$

where U_j represents the available nodal utility, and β is introduced by RS algorithm to hold a tolerance guard, which is tunable. If node decides not to save the received packet, it will proceed to redistribute the packet to other nodes for storage.

7.1.2 Probabilistic Data Redistribution

The reason of redistributing data packet following a probabilistic manner is two fold. First, having packets probabilistically broadcast consume significantly less communication cost for offloading data to other storage nodes than that of gossip flooding. Second, statistically such a probabilistic redistribution can mitigate the risk of data loss despite the fact part of delivery paths may fail in the course of redistribution.

Two main challenges arise in design. First is about how to make data redistributed towards reliable storage nodes instead of nodes with higher failure probability. Due to heterogeneous node densities, rebroadcast message can be heavily stored in nodes with denser vicinity but unreliable nodes. Data is unlikely to be redistributed to other nodes, which might have better connection to sink node for collection purpose. *RS* integrates the adaptive power control to change the outcome of data redistribution towards more reliable nodes.

Second challenge is to reduce data redundancy in redistribution. More redundant data packets broadcast in the network implies more energy are consumed to achieve this process. Although energy cost for preserving data persistence is the extra overhead must paid, this

cost needs to be carefully conserved. Excessive waste of energy severely causes nodes to fail due to power depletion.

Probabilistic data redistribution is articulated in Algorithm 9, which describes the transmission power control based rebroadcast and recoding based data redundancy constraining. When the node with decreasing storage space decides not to save the incoming packet, it starts to redistribute the data to other storage nodes with adaptively controlled transmission power. In line $5-8$ of Algorithm 9, node i selects a certain power level for broadcasting, according to a posterior probability distribution of transmission power, i.e. $\{p_1, p_2, ..., p_m\}$. This power distribution is dynamically updated based on available storage space among neighbor nodes. This rebroadcast is invoked when the message is not accepted by the node. Since it is the source of data redistribution, there is no duplicate copy of this packet in the network yet.

Then the rebroadcast packets are treated differently. When a rebroadcast packet arrives at a node with sufficient storage, the message will be accepted and stored (line 11-13). In the other hand, when no sufficient storage is available, receiving node will operate recoding before it rebroadcast out the message with a selected probability. In line 14, it examines the code distance between received packet κ and encoded symbols in the storage space with the same code degree. If there exists data of the same degree which has different code symbols, the packet will replace the one with smallest coding distance from other symbols with a probability α. The purpose of replacement is to maximize the decoding probability, since the encoded symbol with minimum code distance contribute little to the decoding.

In case that no symbol with the same degree is found in the storage, algorithm evaluates the opportunity that rebroadcast packet can be recoded by XOR operation with stored symbols to generate a new symbol. Note that the recoding does not change the code degree associated with the packet (line 23-25). Recoding the same rebroadcast message in different nodes can mix the message randomly with diverse symbols, generating non-duplicate encoded data. Hence, it reduces the data redundancy in rebroadcast messages, avoiding wasting energy in broadcasting duplicate messages.

Algorithm 9 Probabilistic Data Redistribution

Input: *Effective storage space* C_i, β, *max storage* C_{max}, *replacement factor* α, *Power distribution* $\{p_1, p_2, ..., p_m\}$

Draw random variable ρ from range $(0, 1)$
Set $q_i = \beta \cdot U_i = \beta \cdot (\frac{\delta_i \cdot C_i}{C_{max}})$
if $\rho < q_i$ **then**
  Save this data packet to local storage
else
  Node i select power level m with probability p_m
  Rebroadcast the data packet κ using selected transmission power m.
end if
Node j upon receiving a rebroadcast data packet κ
Draw random variable ρ from range $(0, 1)$
if $\rho < q_j$ **then**
  Save the data packet to local storage flash
else
  if Check coding distance$(\kappa, \mathcal{S}) > 0$ **then**
    Select symbol S_l with smallest distance with other symbols in storage
    Replace S_l with κ with probability α
    if ISRECODABLE$(\kappa, \mathcal{S})$ == TRUE **then**
      Recode κ by XORing symbols in storage
    end if
    Rebroadcast packet κ' with probability $(1 - \alpha)$
  else
    if ISRECODABLE$(\kappa, \mathcal{S})$ == TRUE **then**
      Recode κ into κ' by XORing symbols from storage
    end if
    Rebroadcast new packet κ'
  end if
end if

Adaptive power control and *constrained data redundancy* are two major technical building blocks in the probabilistic data redistribution, which are described as follows.

Adaptive Power Control Without adjusting radio transmission power, packets may be trapped in low storage region. Adjusting transmission power moves packet out to the nodes with more sufficient and reliable storage space. The transmission power adjustment procedure itself is probabilistic. An intuitive way for adjusting probability of transmission power m is to make it proportional to the total available nodal utilities $\sum U_j^m$ in node j:

$$p_m = c \cdot \sum U_j^m \tag{7.2}$$

The above equation implies that transmission power that reach larger amount of $\sum U_j^{P_i}$ has a larger probability to be selected for data rebroadcast. However, there exists one drawback of directly using the proportional equation to determine the probability of transmission power. Applying this strategy, each node greedily selects the maximum power to rebroadcast throughout the process, leading to significant message flooding and useless data redundancy, though the local data entrapment is relieved. Figure 7.1 shows an example depicting this problem. In the figure, nodes from A to G select their maximum power respectively, i.e. $m = 3$, for it can reach nodes with most nodal utilities. However, each node, like node F, receives 6 copies of rebroadcast messages, which are mostly duplication. Meanwhile, transmission with power m can consume quadratically more energy than that of power $m - 1$. The energy efficiency can be enhanced by letting each node selects their transmission power conservatively. Meanwhile, we still wish to preserve the data persistence, which means that rebroadcast message is stored by at least one storage node. Adaptive power control has been designed to achieve both of goals.

The probability of selecting transmission power level is redistributed to be exponentially proportional to the total nodal utilities among rebroadcasting node's neighbors.

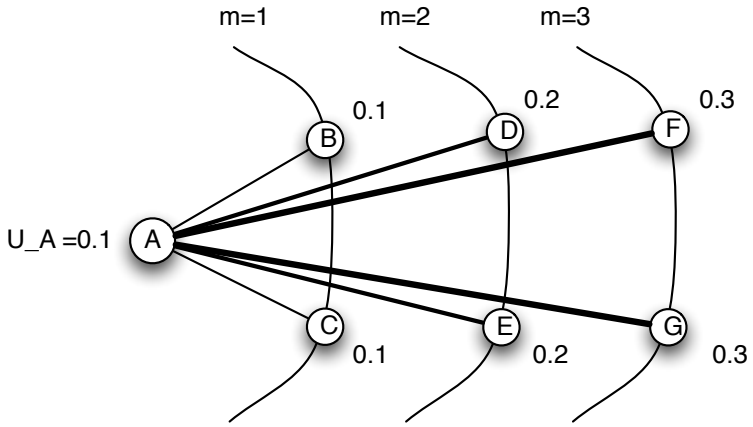


Figure 7.1 Example of transmission power control

$$p_m = (\frac{\bar{U}_j^m}{\bar{U}_j^{m-1}} \sum_j (U_j^m))^\tau \tag{7.3}$$

where τ is the exponential system parameter tunable between 0 and 1 and $\bar{U}_j^m$ is the average nodal utilities of node j and node j's neighbors, when power level m is applied in node j's transmission. When the sum of nodal utilities equals to 1, the expected probability of storing the rebroadcast message is ≥ 1. This ensures that the corresponding power level is selected to guarantee the data persistence. Rather than linear proportional relationship, the above equation has the power exponent $\tau \in (0, 1)$. With system exponent τ, the power level p_m, under which the sum of nodal utilities U_j^m is closest to 1, has highest probability in all power levels. The reasons of adding coefficients $\bar{U}_j^m / \bar{U}_j^{m-1}$ are two fold. First, the power level m, which increases not only total nodal utilities but also the average values, should be given a higher probability in the power distribution. Second, p_m needs to approach to 1 more aggressively once the next power level m leads to a larger average utility amount, so that cut-off power level may appear smaller. A smaller cut-off power level indicates exponential energy conservation. In Equation 7.3, if the calculated $p_m \geq 1$, then let $p_m = 1$. Notice that m, in which $p_m \geq 1$, is the cut-off power level in power probability distribution. Through probability normalization, it ensures that $\sum_m p_m$ equal to 1. The average nodal utility follows that:

$$\bar{U}_j^m = \frac{1}{|N_j^m| + 1} \sum U_j^m$$

In order to calculate the nodal utility summation and average value, node j needs to know its neighbors' utilities. Thus, RS algorithm adds two extra fields in each broadcasting packets: power level indicator and available nodal utility. Upon receiving rebroadcast message, node j records both power level received and available nodal utility from neighbor node.

For instance, in Figure 7.1, before adaptive power control, node A choose power level based on Equation 7.2, so that a total nodal utility can reach or exceed 1. Since $(0.1 + 0.1 * 2 + 0.2 * 2 + 0.3 * 2) = 1.3 > 1$, the most likely power level for node A is $m = 3$. We set $\tau = 0.4$. With adaptive power control, when $m = 1$, it gives $p_1 = (0.1 * 3)^{0.4} = 0.618$. Set $m = 2$, then $p_2 = (\frac{0.14}{0.1} \cdot 0.7)^{0.4} = 1$. Thus, node A's cut-off power is reduced from level 3 to level 2.

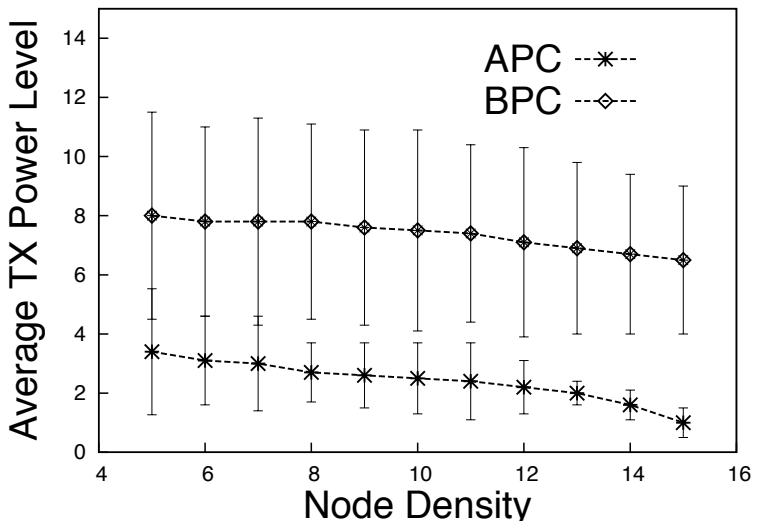


Figure 7.2 Average TX power level for data redistribution.

Figure 7.2 shows the average TX power levels selected by Equation 7.2 (BPC) and Equation 7.3 (APC) under different node densities respectively, within the TOSSIM [69] network simulator. From the results, it shows that adaptive power control (APC) reduces the power level by 60% on average compared with basic power control (BPC). In addition,

the smaller deviation of power level among nodes indicate that energy consumption of TX power is relatively more balanced if executed by *APC*. Moreover, the value of power level decreases slowly in *BPC*. The reason is that each node greedily selects its power level, so that storage space is consumed faster than necessary, making other nodes have to take higher transmission power to discover more storage nodes.

Data Redundancy Constrain Adaptive power control (APC) reduces the energy consumption of rebroadcast as well as the total amount of rebroadcast data replication. However, redistribution packets consist of duplicate data due to the rebroadcasting process. Data content redundancy wastes communication energy as well as the storage space. Therefore, we further constrain the content redundancy of rebroadcast message by conducting symbol recoding without impairing the original code degree distribution.

Recoding procedure is shown in Algorithm 10. Recoding step takes as input the rebroadcast encoded packet S, a set X of both decoded symbols and encoded packets of degree less than 2 and regenerates a fresh encoded packet with the same degree as d. This recoding step mixes the encoded symbol of rebroadcast packet with local symbols in storage, such that multiple copies of broadcast packets can be stored with distinct encoding contents after their redistribution process.

Algorithm 10 Data Recoding

Input: *Received rebroadcast packet S, a set X of both decoded symbols and encoded packets of degree* ≤ 2

Output: *Recoded packet Y*

1: **for all** $s \in S$ **do**
2: $\mathcal{B} \leftarrow \emptyset$
3: **for all** $x \in X$ **do**
4: $\mathcal{B} \leftarrow x, if\{x \Leftrightarrow s\&freq(x) < freq(\mathcal{B})\}$
5: **end for**
6: **if** $\mathcal{B} \neq \emptyset$ **then**
7: $Y \leftarrow S \oplus x$
8: **end if**
9: **end for**

The basic operation in network erasure coding is *XOR*, which is effective and of low computation cost. It verifies the properties that swapping symbol only requires another *XOR* operation. For instance, a native symbol s_1 in S can be replaced with s_2 if target pair $(s_1 \oplus s_2)$ is available, where s_2 is not contained in S. That is, $(s... \oplus s_1) \oplus (s_1 \oplus s_2) = (s... \oplus s_2) = S'$. We denote this relationship as "$s_1 \Leftrightarrow s_2$". Since there are more than 50% encoded packet of degree equals to or less than 2 in LT Codes and Raptor Codes and the search for degree 2 packets are faster than higher degree packets, we mainly leverage those local packets with degree $d \leq 2$ to recode rebroadcast packets. When multiple target pairs are found in the storage or can be generated from native symbols, like $(s_1 \oplus s_3)$ and $(s_1 \oplus s_4)$, the symbol s_i, with less appearance frequency is selected.

Illustrated in Figure 7.3, conduction of symbol recoding constrains the data redundancy. In Figure 7.3, packet of $\{x1 + x2 + x3\}$ is broadcast from node C, and two duplicate copies are received by node A and B respectively. There exist different groups of encoded symbols previously cached in node A and B storage. For example, node A has 3 decoded native symbols $x3, x4, x6$ (degree 1), which has different appearance frequency in the prior symbol recoding, shown in number marked in the right-top corner. According to line 4 in Algorithm 10, node A selects $\{x2 + x5\}$ to recode, because $x5$ satisfies "$x5 \Leftrightarrow x2$" and $\{x2 + x5\}$ has the lower frequency than $\{x3\}$ and $\{x4\}$ as well. The recoded packets from node A and B become distinct and carry different native symbols.

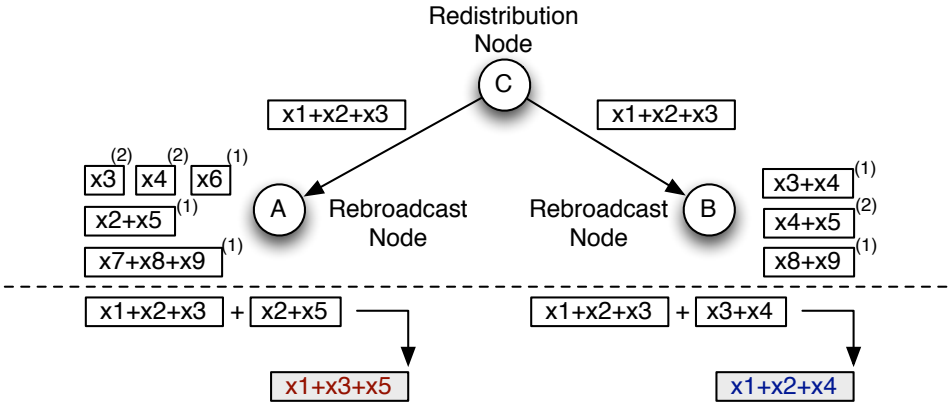


Figure 7.3 Illustration of symbol recoding in rebroadcast node.

Although a successful decoding probability depends on the code degree distribution adopted during encoding, symbol recoding does increase the decoding probability. From LT encoding point of view, our data redundancy constrain approach builds stronger and more reliable connections between left and right components in the Tanner Graph. Introducing more connections between symbol and encoded packet prevent decoding from early failure, in the case of encoded packet loss.

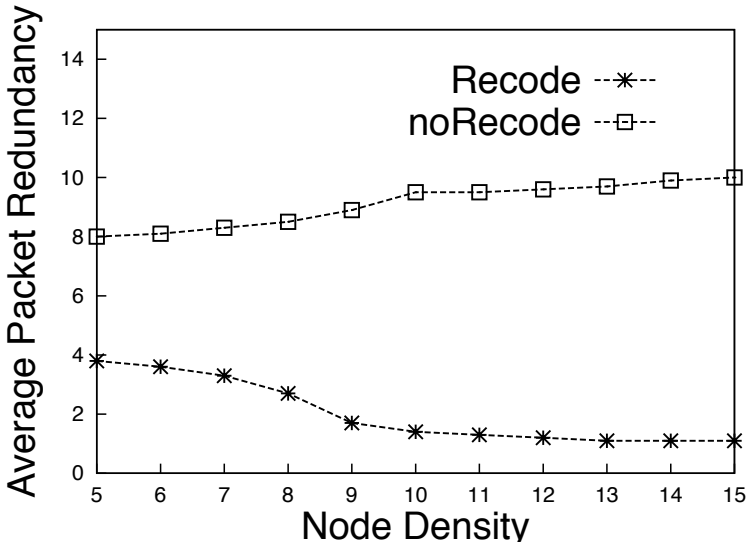


Figure 7.4 Data redundancy under various node densities.

Figure 7.4 evaluates the data redundancy against different node densities. With our recoding on rebroadcast packet, the redundancy of data content has been reduced by at least 50%. With node deployment becomes denser the data redundancy with recoding can be further lowered to approach to 1, which means that there is almost no identical rebroadcast message on the network. In contrast, without recoding the data redundancy increases as the node density, because more nodes are involved in packet rebroadcasting, causing more duplicate packets in the network.

7.1.3 Algorithm Analysis

RS preserves data when disruptive connection causes buffer to overflow. This data quality preservation layer is built upon opportunistic in-network coding-based (*OnCode* [78])

data delivery. The packets delivered in the network before redistribution are handled by *OnCode* protocol. The deliverable symbol size B for unit time span is based on the disruption probability distribution Π, shown in [78].

Lemma 33 *[78] Given a disruption probability distribution* $\Pi = \{\pi_{w_1}, \pi_{w_2}, ..., \pi_{w_n}\}$*, with* π_{w_i} *denoting the probability for discretized disruption probability* w_i*, then by recursive hitting time estimation, the deliverable symbol size* $B = \sum_{i=1}^{n} B_i = \sum_{i=1}^{n} \frac{R_i}{(1+\epsilon)\lambda} = \sum_{i=1}^{n} \frac{1/(1+\epsilon)\lambda}{L(i,sink)} = \sum_{i=1}^{n} \frac{1/(1+\epsilon)\lambda}{\sum_{k \in Nbr(i)} P_{ik}(T_{ik}+L(k,sink))}$*, where both* P_{ik} *and* T_{ik} *are determined by distribution* Π*.*

Hence, the performance bound of *OnCode* is described as:

Theorem 34 *[78] Given the size of symbol set as* B *and degree distribution of Raptor Codes* $\omega(*)$*, the delivery ratio of OnCode is* OPT/λ*, where* $\lambda = \frac{1+\sqrt{1+4\cdot(1-e^{-\epsilon B}/\delta)}}{2\cdot\omega(1)\cdot(1+\epsilon)}$ *(*$\delta, \epsilon \in (0, 1]$*), with a successful decoding probability of all symbols* $\geq (1 - e^{-\epsilon B})$*.*

[78] shows that *OnCode* provides a OPT/λ performance bound. We continue to consider the optimal solution of data preserving in the course of disruptive network connection: OPT. A stable path from distribution node to storage node is always perceived in OPT for distributing data, and exact one copy of data distribution is sufficient for preserving persistence. Therefore, the storage cost for OPT is $O(1)$. For *RS*, adaptive power control in section 7.1.2 ensures that in each broadcast the expected acceptance probability of distribution data in storage node approximates 1, and *RS* adopts 2-hop rebroadcast to make sure the event that storage nodes accept distributed data with probability equivalent to 1.

We assume that there are available B storage space for B innovative data symbols. OPT can exactly leverages all of these space to store redistribution data for persistence. Then, the delivery ratio of *RS* is derived as follows.

Theorem 35 *Given the size of source data set* B*, and degree distribution of Raptor Codes* $\omega(*)$*, the final delivery ratio of RS:* $\eta = OPT \cdot \frac{\ln(\frac{B}{\delta})}{2\lambda}[1 + \omega(1) \cdot \frac{1}{B} + 2\omega(2) \cdot \frac{B-2}{\binom{B}{2}}]$*, where* $\lambda = \frac{1+\sqrt{1+4\cdot(1-e^{-\epsilon B}/\delta)}}{2\cdot\omega(1)\cdot(1+\epsilon)}$ *and* $\delta, \epsilon \in (0, 1]$*, with a successful decoding probability of* B *symbols* $\geq (1 - e^{-\epsilon B})$*.*

Proof 36 *RS adopts* 2*-hop redistribution to ensure the acceptance of data, therefore only* $\frac{B}{2}$ *can be accepted in storage nodes expectedly without recoding. In other words, the final delivery ratio of RS becomes* $\frac{1}{\lambda \cdot 2}$*. However, RS leverages recoding to eliminate data duplication, so that any* 2 *copies are distinct from each node when they are stored. Therefore, the final delivery ratio is determined by the portion of data that can be recoded shown in Algorithm 9.*

Only packets of degree 1 *and* 2 *are utilized in recoding. And we define the event A as that packets are recodable when there exists a symbol x in the packet that can find* $x \Leftrightarrow y$ *in the storage symbol set of either degree* 1 *or* 2*. The probability of event A is* $\ln(\frac{B}{\delta}) \cdot (\omega(1) \cdot \frac{1}{B} + \omega(2) \cdot \frac{B-2}{\binom{B}{2}})$*. If this event occurs, the corresponding delivery ratio of RS is* $1/\lambda$*, otherwise* $1/2\lambda$*. Therefore, the final expected delivery ratio equals to:* $OPT \cdot \frac{\ln(\frac{B}{\delta})}{2\lambda}[1 + \omega(1) \cdot \frac{1}{B} + 2\omega(2) \cdot \frac{B-2}{\binom{B}{2}}]$.

7.2 Performance Evaluation

In this section we present the performance evaluation of RS on TOSSIM [69] simulation. The performance of RS is compared against other existing data collection and redistribution algorithms from the literature. The experimental results demonstrate that data delivery ratio under disruptive networks is substantially improved by our *RS* algorithm, with an efficient energy cost and moderate storage cost. Experimental setup and result analysis are discussed as follows.

7.2.1 Experimental setup

To illustrate the advantages of *Ravine Streams* in improving data delivery ratio, hence enhancing data persistence over disruptive sensor networks, this work implements and compares *RS* with three other existing data collection and redistribution protocols: *EDR2* [55], *PoF* [53] and *NoCode*. *EDR2* is proposed to optimally solve the data redistribution problem under intermittent network connection, i.e. minimizing the data redistribution and retrieval cost. *EDR2* is implemented in a distributed manner by "push-relabel" network flow methods. *PoF* devises an index based potential field, which is used to determine the amount of data redistributed to storage nodes. It shows the optimality of data redistribution without

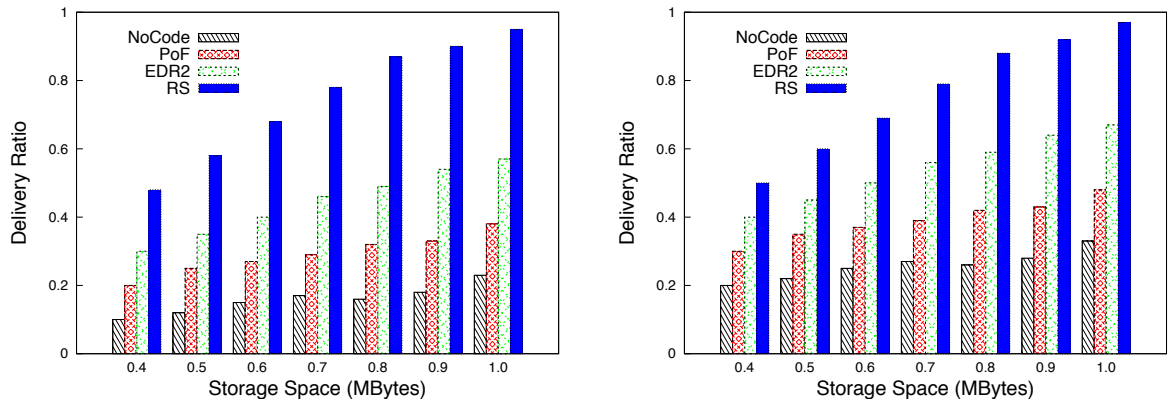
data retrieval cost. *NoCode* is a plain data collection protocol, without specific redistribution and coding strategy. Once node disruption happens, data is redistributed randomly to other storage nodes by broadcasting.

The experiment is carried out in $100m \times 100m$ square area with nodes randomly deployed. We test the algorithms in a network of size 100 and 500 respectively, and the total storage space, nodal failure probability and number of data generators inside network vary to compare the performances under different scenarios. Moreover, energy cost and flash storage cost are also evaluated among different algorithms. For *RS*, the total power levels are set as 8, each of which can reach areas of different radii. Set nodal utility factor $\beta = 0.9$, and replacement factor $\alpha = 0.5$. We show the experimental results in the following sections.

7.2.2 Data persistence under disruptive networks

For numerical evaluation, we use data delivery ratio to represent data persistence. Data delivery ratio is defined as the ratio between the amount of data sent and the actual amount of data recovered by the destinations. Figure 7.5 shows the data delivery ratio for multiple algorithms under variable total storage spaces. The average nodal failure probability is 20% and there are 50 and 200 data generators in network size of 100 and 500 respectively. Each of data generators has $1kbps$ data rate. In Figure 7.5, y-axis denotes the percentage of data sent which can be preserved throughout the network disruption, with x-axis showing different total storage space in nodes. *RS* outperforms other algorithms for all cases of different storage spaces. From both of experiments, the ratio of data persistence improves along with the increasing storage space. For example, it can be observed that with increasing nodal storage space ($0.4M$bytes to $1.0M$bytes), the ratio of data persistence is significantly improved by RS, increasing from 50% to 93%, which is much more than the increments in other algorithms. Compared with the *EDR2* and *PoF*, the application of network erasure coding in *RS* makes packet redistribution more robust. Unlike the *EDR2* and *PoF*, which does not has reliable data redistribution to storage node, *RS* adopts probabilistic broadcasting together with randomized recoding for data redistribution. Therefore, more data packets are preserved even

in the present of disruptive nodes and network conditions. Compared with the *NoCode*, *RS* eliminates the duplicate packet retransmissions. The recoding of redistributed packet with innovative data in *RS* is beneficial for mitigating the data content redundancy. Each packet carries different innovative symbols which contribute to the final decoding. Recoding not only reduces the data redundancy, but also promotes the randomness in mixing symbols. The network erasure coding based data delivery can maximize its coding gains when the symbol selection is pure random across the entire source symbol set.

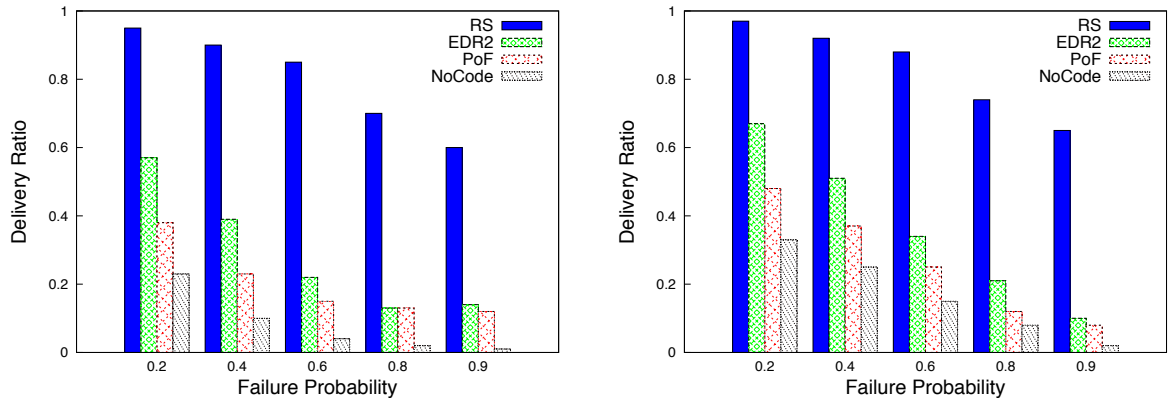


(a) Network Size 100 (b) Network Size 500

Figure 7.5 Data delivery ratio over different storage spaces (Node Failure Probability is 20%).

Figure 7.6 demonstrates the delivery ratio under different node failure probabilities. The storage space of each node is set to $1M$bytes. In the left figure, it shows that RS stills has the best performance in terms of delivery ratio. More importantly, when average nodal failure probability increases, the delivery ratio of *RS* only reduces by 30%, and still maintains above 60% delivery ratio under 90% failure probability. On the contrary, the delivery ratio of *EDR2* and *PoF* fall below 20%, and *NoCode* has even less than 5% data delivery ratio. The right figure in Figure 7.6 demonstrates the same trend. An essential reason is that algorithms other than *RS* fail to consider disruptive network condition during data distribution, and hence algorithms are not able to adapt to intermittent connection and disruptive communication link. For example, after redistribution node in *PoF* determines how many amount of data should be rerouted to the destination storage node, based on

the potential index, redistribution nodes take it for granted that the redistribution data can reach or be stored in the destination. Unfortunately, it is not always true, especially in disruptive networks where the intermittent connection can happen any time during the entire data delivery. This experiment manifests that disruptive communication poses a big challenge to existing data redistribution protocols.



(a) Network Size 100 (b) Network Size 500

Figure 7.6 Data delivery ratio over different failure probabilities (storage=1MByte).

Figure 7.7 demonstrate the delivery ratio under different data generators in the network. With increasing number of data generator, the network traffic becomes heavier and preserving data persistence is hence more challenging. The figure shows that when there are 100 data generators in a network of 500 nodes, *EDR2* and *PoF* can still preserve more 50% data, with 45% data persistence for *NoCode*. However, the data delivery ratio of all three approaches drop significantly as the number of data generator increases from 100 to 350. In the worse case, all three data preserving algorithm only manage to preserve less than 20% amount of innovative data. By comparison, it shows the advantages of *RS*, which retains a steady performance of above 90% data persistence across different number of data generators. Efficient data duplication avoidance (by recoding) and message redundancy reduction (by opportunistic rebroadcasting and adaptive power control) in *RS* play an important role in maximizing data delivery ratio.

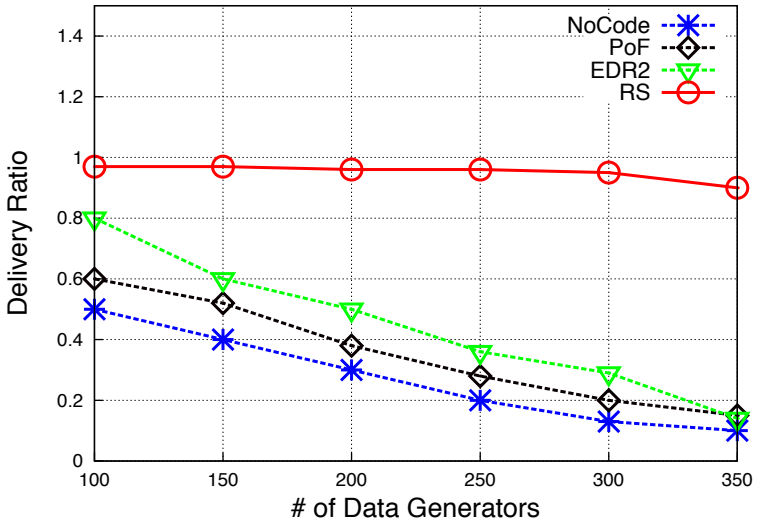


Figure 7.7 Data delivery ratio v.s. amount of data generators (storage=1MByte, failure prob. = 20%).

7.2.3 Storage cost

We evaluate the storage cost for each of the algorithms in Figure 7.8. We observe that *NoCode* method consumes more storage space than others when number of data generator increases higher than 100. Under 350 data generators, there are only 6% storage space on average is available in each of nodes. This implies that the uncontrolled rebroadcasting of *NoCode* results in much redundant packets which requires much more storage space. Although *RS* consumes less storage space than *PoF* and *EDR2* in the network of light traffic, *RS* requires more storage space as the network traffic becomes higher. This is because *RS* adopts the opportunistic rebroadcasting to accommodate the disruptive communication link which introduces moderate duplicate messages. *PoF* and *EDR2* consume less storage space but introduce much more energy cost.

7.2.4 Energy cost

Energy cost is critical to the data preserving, since excessive energy consumption will increase the nodal failure probability due to energy depletion. Illustrated in Figure 7.6,

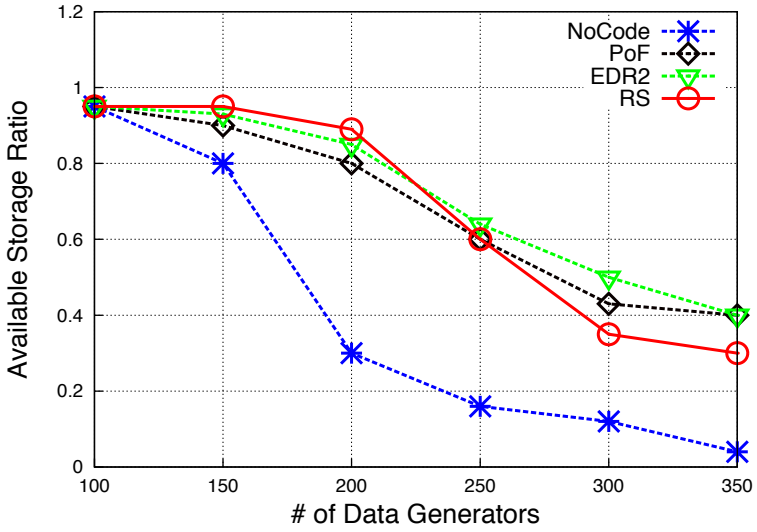


Figure 7.8 Available storage space ratio v.s. amount of data generators (storage=1MByte, failure prob. = 20%).

all the algorithms experience a performance degradation under higher failure probability. Therefore, energy conservation is not only for the sake of efficiency, but also for maximizing data persistence.

The energy cost increment is exponentially proportional to transmission power level. Thus, we evaluate the experimental output of power level adaptation algorithm in Figure 7.9. Figure 7.9 represents the transmission power probability distribution according to Equation 7.3. The diamond-dotted curve is the cumulative probability of power selection. It shows that more than 70% probability fall into power level 3 and 4, which are low-moderate power level for the radio. This results is consistent with the preliminary results from Figure 7.2. It also depicts that adaptive power control can adapt to network dynamics, including neighbor node failure, node storage space changes and node density alteration, with small transmission power cost.

Figure 7.10 demonstrates total energy consumption of different algorithms, normalized against the *Optimal* solution. The optimal solution is assumed to establish a stable delivery path from distribution node to storage node, which only consumes minimum energy cost. In low failure probability, *RS* only expend 2.3 times as *Optimal* solution. It is due to that

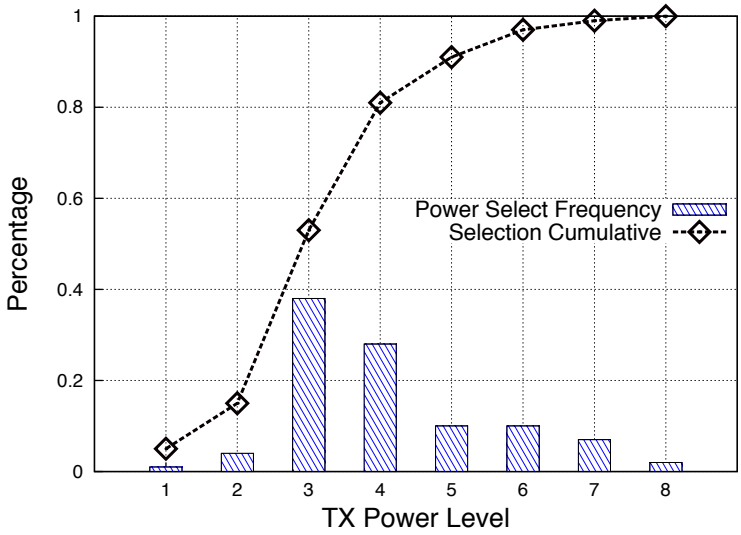


Figure 7.9 Transmission Power Level Distribution.

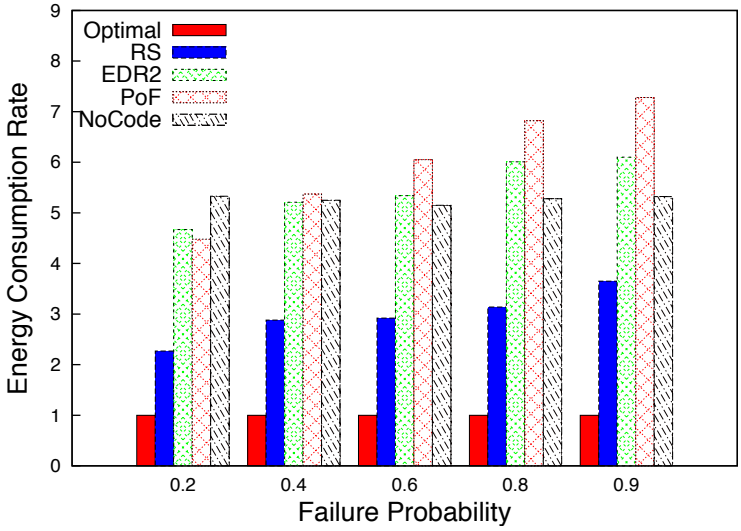


Figure 7.10 Energy Consumption Rate (storage=1MByte).

in *RS* probabilistic rebroadcasting is applied to accommodate the disruptive connection. Compared with *Optimal*, more redundant nodes are required to rebroadcast packets in the absence of global knowledge of node failure prediction. In the worse case, *RS* consumes 3.5 times as much as *Optimal* solution. Although *RS* adjusts its transmission power for better expected coverage, its total energy consumption is 40% less than *EDR2* and *PoF*. The packet retransmission in those two approaches consume considerable energy when the path for data redistribution becomes intermittent. *NoCode* randomly rebroadcasts the packet for redistribution, which is unaware of network dynamics, so that energy consumption of *NoCode* remains 5 times as much as the *Optimal* solution. But the data preserving performance of *NoCode* is dramatically influenced by failure probability as in Figure 7.6.

CHAPTER 8

CONCLUSIONS

This doctoral thesis proposes a set of algorithm based on network erasure coding theory to improve the data quality and preserve data persistence in disruptive wireless sensor networks. Max-flow min-variance algorithm schedule flows among multiple energy-synchronized data paths without coding techniques. It established a benchmark for optimal solution. Due to network dynamics and heterogeneous disruption across network, opportunistic routing and coding is studied to practically preserve data persistence. ONEC lays a theoretic foundation for later three works, with respect to the degree deconvolution and applying erasure codes to network codings. The coding overhead is minimized because the advantageous statistical property of erasure codes is well preserved in our proposed network codings. And OnCode is proposed to provide good quality of data service under the constraints of energy synchronization. The Ravine Streams simultaneously consider the constrained of energy and limited storage space in disruptive sensor networks. ECPC addresses the problem of data storage when sink node becomes unavailable.

REFERENCES

[1] R. Huang, W.-Z. Song, M. Xu, N. Peterson, B. Shirazi, and R. LaHusen, "Real-World Sensor Network for Long-Term Volcano Monitoring: Design and Findings," *IEEE Transactions on Parallel and Distributed Systems*, 2011.

[2] Y. Gu, T. Zhu, and T. He, "ESC: Energy Synchronized Communication in Sustainable Sensor Networks," in *The 17th International Conference on Network Protocols*, Oct. 2009.

[3] S. Biswas and R. Morris, "ExOR: Opportunistic Multi-hop Routing for Wireless Networks," in *Proceedings of the Special Interest Group on Data Communication (SIGCOMM'05)*, ser. SIGCOMM '05, vol. 35, no. 4. Philadelphia, Pennsylvania, USA: ACM, Oct. 2005, pp. 133–144. [Online]. Available: http://dx.doi.org/10.1145/1080091.1080108

[4] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing." in *ACM SIGCOMM*, Aug. 2007.

[5] D. Koutsonikolas, C.-C. Wang, and Y. C. Hu, "CCACK: Efficient Network Coding Based Opportunistic Routing Through Cumulative Coded Acknowledgments," in *IEEE INFOCOM*, Mar. 2010.

[6] R.-S. Liu, P. Sinha, and E. C. Koksal, "Joint Energy Management and Resource Allocation in Rechargeable Sensor Networks," in *INFOCOM 2010*, Mar. 2010.

[7] S. Chen, Y. Fang, and Y. Xia, "Lexicographic Maxmin Fairness for Data Collection in Wireless Sensor Networks," *IEEE TRANSACTIONS ON MOBILE COMPUTING*, vol. 7, pp. 762–776, Jul. 2007.

[8] K.-W. Fan, Z. Zheng, and P. Sinha, "Steady and Fair Rate Allocation for Rechargeable

Sensors in Perpetual Sensor Networks," in *SenSys 2008*, Raleigh, North Carolina, USA, Nov. 2008.

[9] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis, "Interference Aware Fair Rate Control in Wireless Sensor Networks," in *Proceedings of the Special Interest Group on Data Communication (SIGCOMM'06)*, Sep. 2006.

[10] J. Paek and R. Govindan, "RCRT: Rate-Controlled Reliable Transportfor Wireless Sensor Networks," in *Proceedings of the ACM conference on Embedded network sensor systems (SenSys '07)*, Nov. 2007.

[11] B. Hohlt, L. Doherty, and E. Brewer, "Flexible power scheduling for sensor networks," in *Proceedings of the third international symposium on Information processing in sensor networks (IPSN)*, 2004.

[12] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *The 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[13] M. Buettner, G. Yee, E. Anderson, and R. Han, "X-MAC: A Short Preamble MAC Protocol For Duty-Cycled Wireless Sensor Networks," in *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys'06)*, Boulder, Colorado, USA, Nov. 2006.

[14] Y. Sun, O. Gurewitz, and D. B. Johnson, "RI-MAC: A Receiver Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Load." in *Proc. 6th ACM conference on Embedded networked sensor systems (SenSys)*, Nov. 2008.

[15] Y. Gu and T. He, "Bounding Communication Delay in Energy Harvesting Sensor Networks," in *ICDCS*, Jun. 2010.

[16] R. G. Gallager, *Low-Density Parity-Check Codes.* MIT Press, 1963.

[17] M. Luby, "LT Codes," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2002.

[18] A. Shokrollahi, "Raptor Codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[19] X. Jiang, J. Polastre, and D. Culler, "Perpetual Environmentally Powered Sensor Networks," in *IPSN/SPOT 2005*, UCLA, Los Angeles, California, USA, Apr. 2005.

[20] C. Park and P. H. Chou, "AmbiMax: Autonomous energy harvesting platform for multi-supply wireless sensor nodes," in *SECON*, Sep. 2006.

[21] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. Srivastava, "Design considerations for solar energy harvesting wireless embedded systems," *IEEE SPOTS*, 2005.

[22] P. Dutta, J. Hui, J. Jeong, S. Kim, C. Sharp, J. Taneja, G. Tolle, K. Whitehouse, and D. Culler, "Trio: Enabling Sustainable and Scalable Outdoor Wireless Sensor Network Deployments," in *IPSN2006*, Nashville, Tennessee, USA, Apr. 2006.

[23] J. Jeong, X. Jiang, and D. Culler, "Design and analysis of micro-solar power systems for Wireless Sensor Networks," in *INSS 2008*, Kanazawa, Japan, Jun. 2008.

[24] T. Zhu, Z. Zhong, Y. Gu, T. He, and Z.-L. Zhang, "Leakage-Aware Energy Synchronization for Wireless Sensor Networks," in *MobiSys 2009*, Krakow, Poland, Jun. 2009.

[25] M. Gorlatova, P. Kinget, I. Kymissis, D. Rubenstein, X. Wang, and G. Zussman, "Challenge: Ultra-Low-Power Energy-Harvesting Active Networked Tags (EnHANTs)," in *MobiCom*, Sep. 2009.

[26] M. Gorlatova, A. Wallwater, and G. Zussman, "Networking Low-Power Energy Harvesting Devices: Measurements and Algorithms," in *IEEE INFOCOM*, Apr. 2011.

[27] Y. Yang, L. Wang, D. K. Noh, H. K. Le, and T. F. Abdelzaher, "SolarStore: Enhancing Data Reliability in Solar-powered Storage-centric Sensor Networks," Jun. 2009.

[28] Y. Yang, L. Su, Y. Gao, and T. F. Abdelzaher, "SolarCode: Utilizing Erasure Codes for Reliable Data Delivery in Solar-powered Wireless Sensor Networks," Mar. 2010.

[29] J. Le, J. C. S. Lui, and D.-M. Chiu, "DCAR: Distributed Coding-Aware Routing in Wireless Networks," *IEEE TRANSACTIONS ON MOBILE COMPUTING*, vol. 9, no. 4, Apr. 2010.

[30] Y. Yan, B. Zhang, J. Zheng, and J. Ma, "Core: A coding-aware opportunistic routing mechanism for wireless mesh networks," *IEEE Wireless Communications*, vol. 17, no. 3, pp. 96–103, Jun. 2010.

[31] S. Puducheri, J. Kliewer, and T. E. Fuja, "Distributed LT Codes," in *International Symposium on Information and Theory*, Sep. 2006.

[32] ——, "On the Performance of Distributed LT Codes," in *Allerton Conf. Communications, Control and Computing*, Sep. 2006.

[33] M.-L. Champel, K. Huguenin, A.-M. Kermarrec, and N. Le Scouarnec, "LT Network Codes," in *ICDCS*, Jun. 2010.

[34] Y. Lin, B. Liang, and B. Li, "SlideOR: Online Opportunistic Network Coding in Wireless Mesh Networks," in *Mini-Conference at IEEE INFOCOM 2010*, Mar. 2010.

[35] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein, "Growth Codes: Maximizing Sensor Network Data Persistence," in *ACM SIGCOMM*, Sep. 2006.

[36] M. Xu, W.-Z. Song, and Y. Zhao, "Opportunistic Network Erasure Coding in Disruptive Sensor Networks," in *The 8th IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (IEEE MASS'11)*, 2011.

[37] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Ubiquitous Access to Distributed Data in Large-Scale Sensor Networks through Decentralized Erasure Codes," in *4th International Symposium on Information Processing in Sensor Networks (IPSN)*, Apr. 2005.

[38] ——, "Distributed fountain codes for networked storage," in *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2006.

[39] Y. Lin, B. Liang, and B. Li, "Data Persistence in Large-scale Sensor Networks with Decentralized Fountain Codes," in *Proceedings of the 26th Annual IEEE International Conference on Computer Communications (IEEE INFOCOM)*, May 2007.

[40] S. A. Aly, Z. Kong, and E. Soljanin, "Fountain codes based distributedstorage algorithms for large-scale wireless sensor networks," in *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN)*, Apr. 2008.

[41] ——, "Raptor Codes Based Distributed Storage Algorithms for Wireless Sensor Networks," in *IEEE International Symposium on Information Theory*, Jul. 2008.

[42] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A Survey on Network Codes for Distributed Storage," *Proceedings of IEEE*, vol. 99, no. 3, Mar. 2011.

[43] D. Vukobratovic, C. Stefanovic, V. Crnojevic, F. Chiti, and R. Fantacci, "A Packet-Centric Approach to Distributed Rateless Coding in Wireless Sensor Networks," in *Proceedings of IEEE SECON*, Jun. 2009.

[44] L. Luo, Q. Cao, C. Huang, T. Abdelzaher, J. A. Stankovic, and MichaelWard, "Enviromic: towards cooperative storage and retrieval in audio sensor networks," in *ICDCS*, Jun. 2007.

[45] L. Luo, C. Huang, T. Abdelzaher, and J. Stankovic, "EnviroStore: A cooperative storage system for disconnected operation in sensor networks," in *Proceedings of INFOCOM*, May 2007.

[46] T. Hara and S. K. Madria, "Data Replication for Improving Data Accessibility in Ad Hoc Networks," *IEEE TRANSACTIONS ON MOBILE COMPUTING*, vol. 5, no. 11, Nov. 2006.

[47] A. Oka and L. Lampe, "Data Extraction from Wireless Sensor Networks Using Distributed Fountain Codes," *IEEE TRANSACTIONS ON COMMUNICATIONS*, vol. 57, no. 9, Sep. 2009.

[48] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network Coding for Distributed Storage Systems," *IEEE TRANSACTIONS ON INFORMATION THEORY*, vol. 56, no. 9, Sep. 2010.

[49] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data," in *ACM SIGCOMM*, Sep. 1998.

[50] D. Munaretto, J. Widmer, M. Rossi, and M. Zorzi, "Resilient Coding Algorithms for Sensor Network Data Persistence," in *Proc. EWSN*, 2008.

[51] Y. Lin, B. Liang, and B. Li, "Geometric random linear codes in sensor networks," in *IEEE International Conference on Communications*, May 2008.

[52] M. Albano and J. Gao, "In-Network Coding for Resilient Sensor Data Storage and Efficient Data Mule Collection," *The 6th International Workshop on Algorithms for Sensor Systems, Wireless Ad Hoc Networks and Autonomous Mobile Entities*, Jul. 2010.

[53] B. Tang, N. Jaggi, H. Wu, and R. Kurkal, "Energy-Efficient Data Redistribution in Sensor Networks," in *MASS*, Aug. 2010.

[54] M. Takahashi, B. Tang, and N. Jaggi, "Energy-Efficient Data Preservation in Intermittently Connected Sensor Networks," in *The Third International Workshop on Wireless Sensor, Actuator and Robot Networks*, Apr. 2011.

[55] M. Valero, M. Xu, N. A. Mancuso, W.-Z. Song, and R. Beyah, "EDR2: A Sink Failure Resilient Approach for WSNs," in *IEEE International Conference on Communications*, Jun. 2012.

[56] J. Liang, J. Wang, X. Zhang, and J. Chen, "An Adaptive Probability Broadcast-based Data Preservation Protocol in Wireless Sensor Networks," in *IEEE International Conference on Communications*, Jun. 2011.

[57] X. Hou, Z. Sumpter, L. Burson, X. Xue, and B. Tang, "Maximizing Data Preservation in IntermittentlyConnected Sensor Networks," in *IEEE International Conference on Mobile Ad hoc and Sensor Systems*, Oct. 2012.

[58] W.-Z. Song, R. Huang, B. Shirazi, and R. Lahusen, "TreeMAC: Localized tdma mac protocol for high-throughput and fairness in sensor networks," in *The 7th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, Mar. 2009.

[59] T. L. Pham, I. Lavallee, M. Bui, and S. H. Do, "A Distributed Algorithm for the Maximum Flow Problem," in *Proceedings of the 4th International Symposium on Parallel and Distributed Computing*, Jul. 2005.

[60] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum flow problem," *Journal of the ACM (JACM)*, vol. 35, pp. 921–940, Oct. 1988.

[61] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. McGraw-Hill Book Company, 2001.

[62] CME6005:http://www.c-max-time.com/products/showProduct.php?id=2.

[63] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, "The Flooding Time Synchronization Protocol," in *Proc. 2nd ACM conference on Embedded networked sensor systems (SenSys)*, Baltimore, MD, USA, Nov. 2004.

[64] G. Lu, D. De, M. Xu, W.-Z. Song, and B. Shirazi, "TelosW: Enabling Ultra-Low Power Wake-On Sensor Network," in *Seventh International Conference on Networked Sensing Systems (INSS'10)*, Kassel, Germany, Jun. 2010.

[65] M. Luby, M. Mitzenmacher, and A. Shokrollahi, "Analysis of Random Processes via And-Or Tree Evaluation," in *Proc. of the 9th Annual SIAM Symp. on Discrete Algorithms (SODA)*, Jan. 1998.

[66] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection Tree Protocol," in *Proc. of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2009.

[67] A. Kamthe, M. A. Carreira-Perpinan, and A. E. Cerpa, "M&M: Multi-level Markov Model for Wireless Link Simulations," in *SENSYS*, Nov. 2009.

[68] H. Lee, A. Cerpa, and Levis, "Improving wireless simulation through noise modeling," in *IPSN*, Apr. 2007.

[69] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications," in *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys'03)*, 2003.

[70] P. Levis, "TinyOS: http://www.tinyos.net."

[71] P. Levis, S. Madden, J. Polastre, R. Szewczyk, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "TinyOS: An Operating System for Sensor Networks," *Ambient Intelligence. W. Weber, J. Rabaey, and E. Aarts (Eds.), Springer-Verlag*, 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.91.6525

[72] M. Penrose, *Random Geometric Graphs*. Oxford University Press, 2003.

[73] E. T. Jaynes, *Probability theory: the logic of science*. Cambridge University Press, 2003.

[74] T. J. Richardson and R. L. Urbanke, "Efficient Encoding of Low-Density Parity-Check Codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, Feb. 2001.

[75] C. Harrelson, L. Ip, and W. Wang, "Limited Randomness LT Codes," in *Proceedings of the 41st Annual Allerton Conference on Communication, Control, and Computing*, 2003.

[76] G. R. Grimmett and D. R. Stirzaker, *Probability and Random Processes*, 3rd ed. Oxford University Press, 2001.

[77] N. Alon and J. H. Spencer, *The Probabilistic Method*. Wiley-Interscience Series in Discrete Mathematics and Optimization, 2000.

[78] M. Xu, W.-Z. Song, and D. Heo, "OnCode: Opportunistic In-Network Coding and Delivery in Energy-Synchronized Sensor Networks," available at http://sensorweb.cs.gsu.edu/~xum/pub/OnCodeTR.pdf, Tech. Rep., May 2012.