INFRASTRUCTURES FOR DATA DISSEMINATION AND IN-NETWORK STORAGE IN LOCATION-UNAWARE WIRELESS SENSOR NETWORKS

BY SILVIJA KOKALJ-FILIPOVIĆ

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Professor Roy D. Yates and Professor Predrag Spasojević

and approved by

New Brunswick, New Jersey

Jan, 2009

ABSTRACT OF THE DISSERTATION

Infrastructures for Data Dissemination and In-Network Storage in Location-Unaware Wireless Sensor Networks

by Silvija Kokalj-Filipović

Dissertation Directors: Professor Roy D. Yates and Professor Predrag Spasojević

For wireless sensor networks with many location-unaware nodes, we propose mechanisms to organize nodes in an infrastructure of intersecting paths, suitable for efficient data dissemination and event localization. As an underpinning for such an infrastructure, we propose a protocol, dubbed BeSpoken, that steers data transmissions along a straight path called a spoke. The BeSpoken protocol implements a simple, spatially recursive process, where a basic set of control packets and a data packet are exchanged repeatedly among daisy-chained relays that constitute the spoke. The protocol directs data transmissions by randomly selecting relays to retransmit data packets from crescent-shaped areas along the spoke axis. The resulting random walk of the spoke hop sequence may be modeled as a two dimensional Markov process. Analysis of this model results in design rules for protocol parameters that minimize energy consumption while ensuring that spokes propagate far enough and have a limited wobble with respect to the axis.

Finally we show how the spokes serve as the building block of an infrastructure that can be used for source localization and data search and dissemination. In particular, we demonstrate how to increase data availability and persistence through the application of distributed coding techniques over concentric circular subnetworks forming the infrastructure. The goal is to allow for a reduced delay collection by a data collector who accesses the circular network at a random position and random time. The storage nodes within the transmission range of the network's relays linearly combine and store overheard relay transmissions using random decentralized strategies. A data collector first collects a minimum set of coded packets from a subset of storage nodes in its proximity and, by using a message-passing decoder, attempts recovering all source packets from this set. Whenever the decoder stalls, a source packet which restarts decoding is polled/doped from its original source node. The random-walk-based analysis of the decoding/doping process furnishes the collection delay analysis with a prediction on the number of required doped packets. The number of doped packets can be surprisingly small when employed with an Ideal Soliton code degree distribution.

Acknowledgements

My thanks go first to my advisors, Roy Yates and Predrag Spasojević, for their consistent support. I learned a lot from both.

Professor Yates relentlessly demanded quality effort and results, in both research methodology, and its presentational aspects, yet he was patient enough to allow me to get used to his criteria, and to adjust my engineering background to the requirements of academia. I am grateful to him for sharing his great research experience, for instilling in me an appreciation for rigorous scientific writing, and for every piece of advice he gave me.

It is a privilege being a student of Professor Spasojevic whose unassuming manner and amicable disposition belies his extraordinary intellect, his elegant yet inclusive teaching methods and his research perseverance and enthusiasm. Under his direction, I learned how to distill the essence of a complex research problem. He also gave me a chance to practice teaching. His guidance and understanding meant a lot to me; I was fortunate to have such an advisor whom I now regard as both a friend and a consummate professional authority.

In chronological order, I would like to thank people who first encouraged me in the pursuit of this dissertation: Dr. Zoran Miljanić, whose commitment to engineering excellence inspired me to continue my education, Ivan Seskar who introduced me to Winlab, and assisted me throughout, Professor Zoran Gajić for being forthcoming and helpful in administrative issues, and Professor Raychaudhuri who was always supportive and accommodating. I also thank him for valuable advice, which had a special impact on me because of the unique blend of his industry experience and his academic acumen, and for being a member of my PhD committee.

I have been fortunate to come across many interesting and inspiring people during my stay in Winlab. It has been a rare privilege working alongside distinguished researchers and technologists like Dick Frenkiel and Prof. Larry Greenstein, from whom I have received great advice and many valuable suggestions. I thank Professor Marco Gruteser for serving as a member of my committee.

I wish to thank Dr. Emina Soljanin, also a member of my committee, with whom I worked on my coding-related research, for helping me recognize potentials of my work and structure my efforts. It was a pleasure working with Dr. Soljanin, not only because she is an experienced researcher but also because of her commendable interpersonal and mentoring skills, precise articulation of ideas, and very insightful and applicable criticism.

Thanks to all WINLAB graduates and students for their help and friendship, including Ivana Marić, Haris Kremo, Joydeep Acharya, Ruoheng Liu, Chandrasekharan Raman, Jing Lei, Hithesh Nama, Jasvinder Singh, Goran Ivković, Xiaojun Tang, Manik Raina, Yao Li, and many others. I am grateful to James for being who he is; I will miss him a lot.

It was difficult to juggle my family obligations with my work, and occasionally to literally squeeze in time to do research. I would not have succeeded without ongoing support of my family. Thanks especially go to my husband for bearing with me through times of doubt and despair, and through many reincarnations of my identity crisis that admittedly accompanied this beautiful journey.

Dedication

To my daughter Isidora

and my son Filip

◇

I know you inherited a passion for learning, and I only hope that your sharp minds will always be inspired by the poetry in your hearts

◇

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation/ Introduction to Location-Unaware Wireless Sensor Networks

Wireless sensor networks (WSN) have emerged as an approach to instrumenting the physical environment. These networks pose a number of unique technical challenges due to the following factors:

Scarce resources: Sensor nodes are simple, battery-run devices with limited data processing, storage and transmission capabilities. It is imperative that they are operated in a manner that conserves battery energy and ensures network longevity [1]. This requires new routing and data dissemination protocols that run on low-power.

Localization: The collected data needs to be related to the location of the event occurrence. WSNs are typically static; however, nodes are typically assumed to be location unaware, randomly scattered over the monitored area, or displaced by environmental factors [2]. The usefulness of sensed data without spatial coordinates may be highly reduced. On the other hand, installing a global positioning system (GPS) receiver on each sensor node may not be a practical solution for most applications, because of the constraints in size and cost of construction of sensor networks. GPS is also a significant power consumer, and infeasible in environments with dense foliage or other clear-sky impediments [3]. This requires developing new energy-aware localization methods.

Unknown data sinks: In WSN, nodes that make observations, known as data sources, are frequently unaware of which data sinks have interest in their observations. Sinks may be scattered across the network, or located in particular geographic regions. In the geocasting problem [4] data needs to be routed to a geographic region instead of a destination node specified by an address. Flooding the whole network is a trivial form of geocasting when sources are completely unaware of sink locations. In this case, the information propagation from the sources to the sinks is referred to as data dissemination, not data routing.

Large network size: WSNs are expected to contain a large number of nodes, several orders of magnitude larger than the existing wireless networks. Scalability is one of the key issues in wireless sensor networks both during deployment as well as during protocol and algorithm development [5]. Unfortunately, it is both expensive and time consuming to deploy large networks solely for the purpose of building a model or testing a protocol for scalability. Simulating and analyzing systems with a large number of sensor nodes scattered randomly is a computational problem. Therefore, there is a need to develop a methodology that creates and validates networks of an arbitrary size.

Communication framework based on collective identity: WSN-specific forms of communication and data processing are largely based on nodes' cooperation and collective behavior. For energy efficiency reasons, the main model of data propagation is hop-by-hop, where nodes are relaying other nodes' data. Another compelling reason for cooperation stems from the wireless multicast advantage [6], the fact that a wireless transmission can be received by all nodes in the transmission range. The concept of using the energy of the overheard transmissions [6] is further extended to accumulative broadcast which allows nodes outside of the transmission range to collect the energy of the unreliably received signal [7], and other applications, such as opportunistic network coding [8].

There is a substantial body of work studying various WSN problems, ranging from capacity issues tackled by information-theoretic approaches [9, 10], energy-efficient routing algorithms [11,12], localization [2], topology control [13,14] and connectivity [15], analyzed using spatial-stochastic, computational-geometric, graph-theoretic, experimental and heuristic methods. However, models and communication frameworks that are accepted and recognized across all those research communities are needed for comparison of results, knowledge sharing, and ultimately better understanding of WSNs.

We next offer a brief review of the existing results related to the outlined problems, and then present a sketch of a holistic approach to solving these problems, which we propose in this dissertation work.

Localization in location-unaware, randomly deployed networks with scarce resources is a difficult problem. Most research on sensor localization exploits distance or angle measurements from anchor nodes (with GPS or preset location), landmarks or neighbors. This is often labeled as computing the virtual coordinates of the nodes, instead of the real ones. These virtual coordinates need not be accurate representations of the underlying geography but, in order to serve as the basis of routing, must capture the underlying relative connectivity between nodes. Several virtual coordinate routing systems are landmark based [16], [17], [18], where a subset of the nodes are selected as landmarks, and every node records its hop-count distances to these landmarks. The landmark distances are then used to generate virtual coordinates for the nodes.

The unknown position of data sinks in WSNs is another important location-related problem that we mentioned. Due to scarce resources and limited processing power of WSN nodes, the unknown position of a data sink makes the task of delivering data especially challenging. Several new communication paradigms, like geocasting, data dissemination and data search, emerged from this problem [5]. We mentioned that geocasting routes data to a geographic region instead of a destination node specified by an address. A trivial form of geocasting is unrestricted flooding, a simple dissemination method that leads to a broadcast storm of redundant transmissions [19], and consumes more resources than necessary [20].

However, there are several dissemination techniques that use flooding selectively. In a push approach [21], a publishing process plants pointers in the network that can be used by the interested sinks to establish a path to the correct source. Publishing mechanisms are largely based on flooding and consequent path endorsement. In the alternative pull approach [22], widely used if the number of sources is much larger then the number of interested sinks, the sinks flood their interests, so that a source can endorse a path toward the sink. Intanagonwiwat et al [22] introduced a data-centric mechanism called *directed diffusion* in which interest requests (queries) are flooded into the network leaving gradient paths back to the sink.

An alternative to flooding is the use of state information stored in selected nodes (possibly along a path) to direct search toward the correct source. In the combined *push-pull* approach, it is the intersection of a dissemination path and a search path that guarantees the success of a search [23]. In [24] the authors propose a push-pull model of data dissemination and gathering, called the “comb-needle.” They analyze the energy consumption of the proposed model, assuming the availability of node position information. *Rumor Routing* [23] introduces the concept of agents, packets that advertise a source’s data along a random walk path that resembles a fairly straight trajectory. The query packet follows a similar random walk path, and the success of the search is based on the high probability that the two sufficiently long lines in a bounded rectangle intersect.

Disseminating data along straight trajectories, studied here, is conceptually closest to geographic greedy forwarding schemes [25,26] used for routing to known geographic destinations, in the sense that instead of greedily approaching the sink, in our approach the data is greedily directed away from the source. The idea of geographic forwarding is so compelling in WSNs that a number of authors have tried to use this approach even when actual node locations are not available, by using virtual instead of geographic coordinates [27,28]. Greedy forwarding, as a simple, efficient and scalable strategy, became a promising routing scheme for large sensor networks. In a geographic greedy forwarding scheme, a source node knows the location of the destination node, either by acquiring it from a location service [29], or by computing it using a hash function in a datacentric storage scheme [30]. A packet is forwarded to a one-hop neighbor which is closer to the destination than the current node. This process is repeated until the packet reaches the destination. Geographic forwarding suffers from the so-called local minimum phenomenon, when all neighbors of the current packet recipient are farther away from the destination than the node itself. To help packets get out of the local minimum, Karp and Kung [25], and independently Bose et al [31], proposed the idea of combining the greedy forwarding and the perimeter routing on a planar graph which describes the connectivity of the original network. Within the data dissemination framework studied here, we also propose a solution for the local minimum problem arising from our forwarding method.

For nodes that are location-aware, either virtually or literally, data dissemination and search algorithms based on likely intersection of message paths could reuse many of the greedy routing mechanisms. For randomly deployed WSNs, without location awareness, and with no infrastructure or landmarks, efficient data search is an unsolved problem. Efficient solutions for data dissemination and search require a good connectivity model, which is a problem by itself, arising from the sheer size of WSNs, randomness of deployment and wireless propagation issues. Two typical models for connectivity have been in widespread use in the sensor network community: unit disk modeling [32] (or Random Geometric Graph) and empirical data traces. It is NP-hard to determine whether a graph, given without geometry, can be represented as a unit disk graph [33]. Thus, results based on empirically obtained connectivity graphs for the location-unaware WSNs cannot be readily compared with results based on unit disk graph models. The approach of using empirical data traces is also difficult and expensive when creating sufficient number of large networks that are properly characterized. New statistical model of lossy links in Wireless Sensor Networks is given recently in [34]. In reality, connectivity is both spatial and temporal random process. Our approach is based on the intuition that stochastic models of WSN connectivity should be used to design dissemination algorithms, while small-scale irregularities (where a specific network realization diverges from the applied stochastic model) should be overcome through heuristic methods.

1.2 Problem Formulation: Minimum-Delay Data Collection in WSNs

The broad motivation for this work arises from the need to create models that describe WSNs, while abstracting the inherent complexity, which would benefit both experimental and theoretical research. Our idea is to create distributed mechanisms that would insert some infrastructure within the WSN, hence making both modeling and controlling/optimizing data collection much easier.

1.2.1 Problem Statement

We consider a randomly-deployed wireless sensor network with location-unaware nodes. Here, sources are network nodes that have some data about observed events. Each source is producing a packet of independent data. Data sources are unaware, ahead of time, of which data sinks have interest in their observations. Moreover, we do not envision sinks to be regular WSN nodes. Regular WSN nodes could be *temporary* data sinks, if located at the network perimeter or other network segments easily accessible by external data collectors. In this case, we refer to them as *storage* nodes, assisting (frequently-mobile) data collectors that are the actual data sinks. We primarily study such data collection. The goal is to design distributed mechanisms that would cultivate the network of n nodes to support efficient collection of data from a set of k randomly positioned sources.

A more formal problem definition involves a a data sink (a *collector*) that appears at a random position, at random time, and aims to collect all the k source data packets. The network's goal is to ensure that the data packets be efficiently disseminated and stored in a manner which allows for a low collection delay upon collector's arrival. This study requires one to address some canonical WSN problems, such as dissemination and localization of data, and distributed data storage. Several papers have recently appeared that propose coding-based solutions for distributed data storage in WSNs [35–40]. Inspired by these approaches, while trying to avoid complexity issues attributed to classical WSN models that they utilize, we first aim to depict and model a self-organized network infrastructure, and then to develop data-collecting solutions based on this model. Inspired by forward-progress routing [26, 41–43], we envisioned the infrastructure as an overlay dissemination network within the WSN, composed of straight propagation paths. Such a propagation path is possible even with location-unaware nodes if the selection of the next forwarding relay is controlled by a protocol, so that the relay always gets picked from an ”innovation” set of the most recently covered nodes. We here introduce a distributed mechanism that organizes sequences of relays in described manner. We consider this mechanism as both a dissemination protocol, and a tool to build an infrastructure of relatively straight paths referred to as *(spokes)* whose direction and length can be learned with moderate effort.

In summary, we intend to create a WSN infrastructure, based on intersecting spokes, that would satisfy several requirements:

- search for particular instances of data, distinguished either by specific data attributes or by the location of the event that produces data; this search is modeled by an intersection of advertising and query spokes, where the intersection refers equally to respective data trajectories crossing each other, and advertising and query designators matching each other,
- event (data source) localization, using the relative hop distances from the set of nearby

spokes, and

- data storage in easily accessible parts of the infrastructure, that guarantees data persistence until a mobile data collector arrives to collect all the data produced by the network sources; for the purpose of distributed data storage, we intend to create such an infrastructure from simple graph structures, to which we could efficiently apply decentralized coding.

1.2.2 Problem Solution Framework

BeSpoken Dissemination Protocol

A wireless dissemination protocol uses a sequence of wireless transmissions that propagate information from the source to the sinks. However, sinks in WSNs are frequently unknown in advance, and information is typically geocasted to a certain network region. We propose a push-pull dissemination model in which sources push data away along straight propagation paths, referred to as *source spokes*, and similar paths, initiated from the sinks and called *sink spokes*, are used to propagate queries until a matching source spoke is intersected.

Existing models for forward-progress routing do not hold for location-unaware wireless networks; the locations are considered known, either through real, or through virtual coordinates [44]. We show that creation of straight propagation paths is possible even for location unaware nodes. Motivated by the radial symmetry of isotropic wireless transmission, we propose to achieve this directional propagation by utilizing the geometry of overlapping transmission ranges centered around two most recent relays. Based on this geometry, we propose a protocol, dubbed *BeSpoken*, which identifies the ”innovation” set of the most recently covered nodes and selects the next forwarding relay from that set. The innovation set area is designed so that the dissemination always attempts making a forward progress with respect to a direction set by the first two relays. We will show that, when some adaptive heuristics are applied, the proposed protocol should exhibit satisfactory performance under anisotropic propagation as well.

As illustrated in Figure 1.1, a source disseminates data advertisements along the source spokes, and a data collector sends a query along its spokes that may intersect the source spokes. Each intersection represents a successful search. The first data collector spoke to reach one of

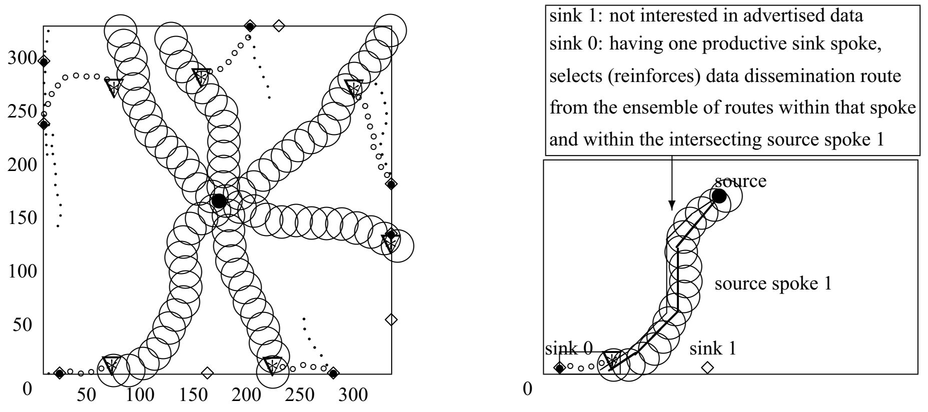


Figure 1.1: The meaning of the name BeSpoken is twofold: the radial lines extending from the source form a pattern that resembles spokes of a wheel and, furthermore, spoke relays *bespeak* the source message. The relative direction of spokes allowing for the wheel pattern is controlled by an extension of the BeSpoken. In this simulation snapshot, source spokes are shown as a sequence of relay transmission *ranges*, to illustrate the fact that each spoke is an *ensemble of possible data routes*. The sequence of wireless transmission *relays* forming a productive sink spoke is denoted by tiny circles (see the boxed spoke in the closeup). Unproductive sink spokes are represented by dots. The search success is marked by a ∇ with an inscribed $*$.

the source spokes is called *productive.* Successful search is to be followed by the endorsement of a route along the intersecting spokes and subsequent data dissemination.

Several papers consider spatial properties of the dissemination route. Different forms of spatially constrained random walks are discussed in [23], [45], [46], while the idea of a trajectory-based dissemination is presented in [47], [48]. None of these dissemination approaches enable unknown source localization. We propose a scheme where application-cognizant data collectors, equipped with GPS and *direction-of-arrival (DoA)* estimation capabilities, can determine the positions of the nodes along the productive sink spoke, and let them know of their positions. The other part of the endorsed route can be learned based on the known position of the intersection nodes, provided that the direction of the source spoke is known. This enables gradual source localization, as described in Chapter 5.

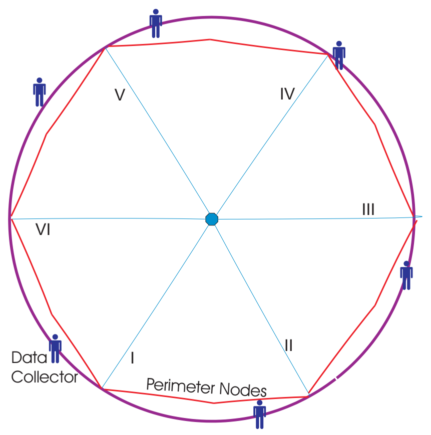
BeSpoken Infrastructure

Source spokes, utilized to convey the information of an event from the source to the areas where a sink is likely to appear, may remain active for a relatively long time period. Following a bootstrap period whose duration depends on the frequency and spatial density of the events in the network, the spoke infrastructure emerges as a system of intersecting paths that completely tessellate the sensor network space. This fairly regular infrastructure provides a way to map subsequent events to areas between the known paths, and to aid efficient navigation toward the associated sources. In this manner, network connectivity graph is compressed into a finite number of intersecting paths obtained by constraining the connectivity conditions. Hence, we could model the network behavior through a simpler graph corresponding to such an infrastructure.

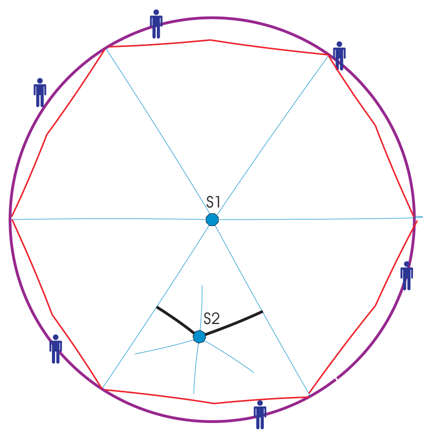
We will show in Chapter 5 how a BeSpoken-enabled WSN can be partitioned into simple circular graphs connected by so-called backbone spokes. This allows easy adaptation of various techniques and models developed for regular graphs and network grids. It is not hard to envision how this framework may bootstrap a number of mechanisms for energy-efficient dissemination, load balancing and rapid data propagation in desired directions. However, in this work we only concentrate on one potential usage model of the BeSpoken infrastructure - efficient data collection by a randomly positioned external collector.

BeSpoken Infrastructure Bootstrap

We assume that there exists a central node (manually placed/ preset landmark, as in [16], [17]) that is responsible for the network infrastructure bootstrapping. The central node extends its equally spaced spokes to the network perimeter, thus swiping the whole network area, as illustrated in Fig. 1.2. We have developed and simulated an extension to the basic BeSpoken that let us create six equally spaced spokes with high probability. The BeSpoken design parameters are preprogrammed for this node to create spokes of sufficient length and straightness, according to the estimated area and network size. We call the spokes of the central node *backbone spokes*. We refer to the areas between the adjacent backbone spokes as wedges. The mobile data collectors are located around the network perimeter (the network area may not be accessible, such as in disaster recovery applications). We further assume that data collectors are more capable in terms of direction-of-arrival estimation, and in being location-aware and application-cognizant. During the infrastructure bootstrap phase they employ their DoA capabilities to infer the direction of the backbone spokes, and hence facilitate the mapping of the infrastructure to geographic coordinates. During the bootstrap phase, the data collectors first create spokes that



(a)



(b)

Figure 1.2: **(a)** Before any event happens, the infrastructure is built around one central node (preset landmark), with the assistance of one or more (over time) external (frequently mobile) data collectors. It consists of equally spaced backbone spokes, and a perimeter route, which partition network area into spoke-delineated wedges. **(b)** The two shortest paths from a source (S_2) to adjacent backbone spokes are the transversal spokes, which most likely form a 120^o angle due to the geometry of the infrastructure

propagate a query for *any* backbone node. Once the query intersects the backbone spoke, the data collector localizes the intersection, then either employs its own mobility or requests the cooperation of the nearby data collectors to poll the nodes along the backbone spoke and estimate the DoA of the response packets. Assuming sufficient density of data collectors along the perimeter, we claim that at the end of the bootstrap phase the directions and lengths of all the backbone spokes are known by all the nodes within the spokes. Alternatively (or jointly), estimating directions of the backbone spokes can be the task of the central node. In addition, the last relay in each backbone spoke extends two spokes in the directions 60^o off the backbone spoke direction. These lateral spokes form the perimeter route of the infrastructure, as shown in Fig. 1.2 (a).

The mobility models of data collectors and data collection frequency ultimately define the usage model of the created infrastructure. We here primarily concentrate on a model where data collectors might be unavailable for long periods of time, but when they do appear, the accent is on collecting data off the closest network perimeter spokes, as opposed to either traversing the network area, or moving around it in order to obtain the data from within the network. Also, this work concentrates on the aspects of the infrastructure that allow collecting all data produced by the network sources.

We assume that there are k independent sources out of n nodes, producing one packet of data each, as in [35–40]. However, this does not preclude supporting other data source models, such as spatially and temporally correlated data. For all practical purposes, correlated data sources can be treated identically assuming some Slepian-Wolf [49] based pre-coding is used.

1.2.3 Problem Solution: Coding for Collection

Given this infrastructure of intersecting spokes, for scalability reasons we partition the network into autonomous parts, or subnetworks, to which we apply coding-based methods for *distributed data storage*. The applied storage protocol should make the entire (possibly encoded) data set available in any collection of $k\,(1+\epsilon)$ storage nodes, accessible by a data collector who approaches the network perimeter at an unknown location (see Figure 1.2). As nodes are unaware of each other, and uncoordinated, a distributed solution for the storage usually requires developing efficient and scalable methods of data *dissemination* from the k sources to *randomly sampled* network nodes (with constrained storage space) [36–40]. Here, simple subnetwork graphs, emerged as a result of infrastructure partitioning, allow us to apply network-coding based dissemination that relies on deterministic packet trajectories, and, hence, allows us to better control communication cost. In addition, the existence of the BeSpoken infrastructure, combined with decentralized data encoding which is congruous to message-passing decoding, allows for a two-phase data collecting strategy that further reduces communication cost.

1.3 Thesis Organization

In the next two chapters we focus on the mathematical model of the BeSpoken and its analysis for the purpose of spoke design. We quantify and illustrate the performance of both the basic and the adaptive protocol variant in terms of their capability to produce sufficiently long and straight enough spokes.

The existing approaches to distributed storage in WSN, and major issues related to it, are discussed in Chapter 4. The scalable data collection architecture and the supporting infrastructure partitioning are described in Chapter 5. The analysis of the proposed collection methods (based on decentralized fountain data encoding in subnetworks) is the subject of Chapter (6). Because of the simplicity and universality of the subnetwork graphs, the techniques analyzed in Chapter 6 can be applied to data collection problems outside of BeSpoken infrastructure.

Chapter 2

BeSpoken: Directional Data Propagation Without Location Information

2.1 System Model

We consider a large wireless sensor network with location-unaware nodes randomly scattered over a disk-shaped area. Uniform spatial distribution of nodes is assumed. We also assume that the physical layer modulation and coding are designed to compensate for short-scale fading effects and, thus, our transmit power requirements depend only on distance-dependent propagation path loss. Even though in a sensor network environment data rates are low relative to the available bandwidth and interference is not a primary issue, still, our protocol mitigates the interference as it always selects only one node to retransmit. We choose simple isotropic propagation model, in order to make the analysis more tractable.

With isotropic propagation, the received power at node v, given that node u is transmitting with power P_t, is

$$p_r(u, v) = K p_t s(u, v)^{-\alpha}, \tag{2.1}$$

where $s(u, v)$ denotes Euclidean distance between the two nodes, K is a constant, and $\alpha \geq 2$ is the propagation loss coefficient. We define the transmission range r as the maximum distance from the source (transmitter) u at which node v can reliably receive a packet, and the received power at distance r is called receiver sensitivity, denoted

$$p_r^s = p_t r^{-\alpha}. \tag{2.2}$$

For this propagation model, the area in which the transmitted packet is reliably received is a

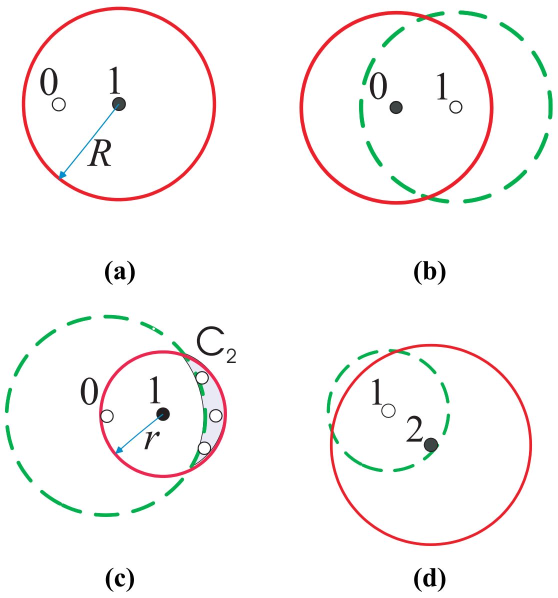


Figure 2.1: BeSpoken Protocol: At each protocol stage, the current transmission range is denoted with the full circle while the previous range is denoted with a dashed circle.

disk of radius r.

In such a WSN, data sources are regular network nodes that make observations. Hence, we assume that data sources are uniformly distributed. The *BeSpoken* protocol organizes a sequence of fixed-power *relay* transmissions that propagate the source message hop-by-hop, without positional or directional information. The hop relays form a *spoke* which may deviate from the radial *spoke axis*. Each spoke hop is organized using a sequence of two control message transmissions followed by the hop data transmission. We use the same transmission power for both data and control packets, but different coding rate and/or modulation format, so that the communication rate for control messages is lower and translates to a longer range.

2.1.1 BeSpoken Protocol

The BeSpoken protocol implements a recursive process illustrated in Figure 2.1 in the following way:

(a) The leading relay (node 1) sends an RTS (request to send) control packet with range

$R = rq$ where $q = 2 - \epsilon$, for small ϵ.

(b) The pivot (node 0) sends a BTS (block to send) control packet with range R.

(c) The leading relay transmits the data packet with range r and becomes the new pivot. The region in which nodes receive this data packet but do not receive the preceding BTS packet forms the *1-st hop crescent* C_2.

(d) A random node from the crescent C_2 becomes the new leading relay by transmitting a new RTS. The process returns to (a) with node 1 as the pivot and node 2 as the leading relay.

This recursive process is initialized by assigning the role of the pivot to the source node which transmits the data packet with a range r. The first node which receives the data packet and gets access to the medium becomes the first leading relay. The underlying ALOHA-type Carrier Sense Multiple Access protocol would resolve any collisions; hence, after a possible additional delay, only one random node from the crescent would transmit the RTS packet.

2.1.2 Problem Formulation

To describe the effects of the data and control ranges r and R, we evaluate the spoke behavior with respect to the constraints:

- **Outage:** the probability that a spoke dies before reaching a distance d is small,
- **Wobbliness:** the deviation of the instantaneous spoke direction with respect to the spoke axis is within defined limits.

The vector from node 0 to node 1 in Figure 2.1 defines the spoke axis. The crescent subtending angle determines how much the spoke may deviate from the spoke axis direction. The parameter $q = R/r$ determines the maximum crescent subtending angle. A large subtending angle fosters wobbliness, yet it implies a larger crescent, which increases chances that a relay will be found to retransmit data. Fixing q to a small value that limits wobbliness requires increasing r to generate a large enough crescent and decrease the outage probability. Note that the energy per hop grows as r^{α}, where $\alpha \geq 2$ is the propagation loss coefficient, so that the total energy

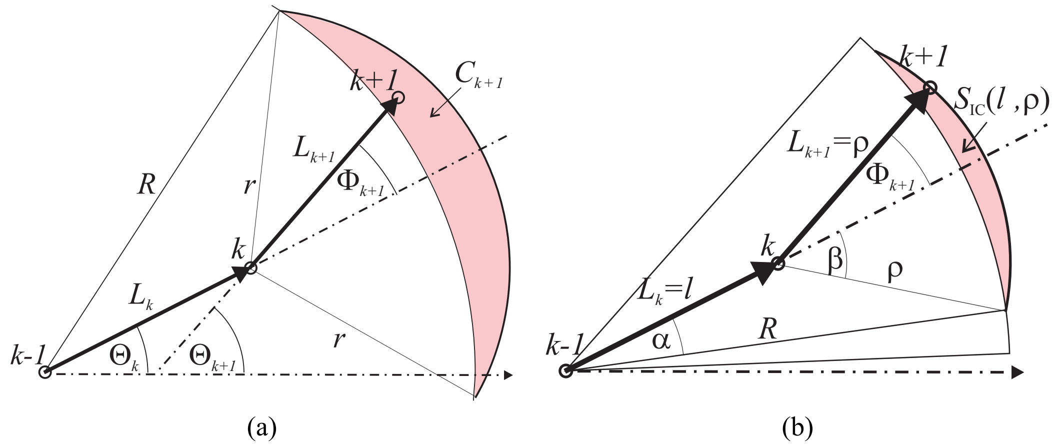


Figure 2.2: **(a)** At hop $k+1$, node $k+1$ is distance L_{k+1} from node k and the current spoke direction is $\Theta_{k+1} = \Theta_k + \Phi_{k+1}$. **(b)** Given $L_k = l$ and $L_{k+1} = \rho$, the angular hop displacement Φ_{k+1} is constrained to the interval $-\beta \leq \Phi_{k+1} \leq \beta$ where the maximum angular displacement at hop $k+1$ is $\beta = \beta(l, \rho)$. The shaded area denotes the interior crescent of area $S_{\text{IC}}(l, \rho)$.

per spoke of length d grows as $dr^{\alpha-1}$. Hence, minimizing the transmission range r corresponds to a minimum energy objective.

These contending tendencies illustrate the importance of the protocol parameters design. In our analysis, we show that outage and wobbliness constraints can be decoupled. Consequently, as a result of the outage constraint analysis, we give the design guidelines for the parameter r. We demonstrate that satisfying the wobbliness constraint requires one to find the minimum q so that the spoke direction is within the limits after η hops, where η is a sufficient number of hops to reach the target distance d, given r. We develop closed-form expressions that serve as bounds for the values of q, ensuring that the wobbliness constraint is satisfied.

2.2 Spoke Modeling

2.2.1 BeSpoken Geometry

Figure 2.2(a) depicts hops k and $k+1$. At the completion of hop k, the length L_k denotes the *current hop length* and the angle Θ_k denotes the *current spoke direction*.

From Figure 2.2(b) we observe that given $L_k = l$ and $L_{k+1} = \rho$ the control circle of radius

R centered at node $k-1$ and the circle of radius ρ centered at node k specify a radius ρ arc for the possible positions of node $k+1$. The endpoints of this radius ρ arc constrain the *angular hop displacement* Φ_{k+1} to the interval $-\beta \leq \Phi_{k+1} \leq \beta$ where the maximum angular displacement is $\beta = \beta(l, \rho)$. Applying the law of cosines to the complementary angle $\pi - \beta(l, \rho)$ yields

$$\cos \beta(l, \rho) \quad = \quad \frac{R^2 - \rho^2 - l^2}{2l\rho}. \tag{2.3}$$

We also observe that the region between the radius R control circle and the radius ρ arc defines an *interior crescent*, shown as the shaded area in Figure 2.2(b). From geometric arguments, it can be verified that the area of this interior crescent is

$$S_{\text{IC}}(l, \rho) = 2\rho^2\beta(l, \rho) - 2R^2\alpha(l, \rho) + Rl \sin \alpha(l, \rho) \tag{2.4}$$

where $\alpha(l, \rho)$ is found from the law of cosines to satisfy

$$\cos \alpha(l, \rho) = (R^2 - \rho^2 + l^2)/(2lR). \tag{2.5}$$

Note that L_{k+1} can vary from a minimum value of $R - L_k$ to a maximum value of r. The induced interior crescent C_{k+1} in Figure 2.2(a) has an area $S_c(L_k) = S_{\text{IC}}(L_k, r)$. We note that C_{k+1}, termed the *current crescent*, is the set of all possible positions of the node $k+1$.

2.2.2 Markov Process Model for Hop Length Evolution

For design purposes we assume that the spatial distribution of network nodes is a planar Poisson point process of intensity $\lambda = 1$. Thus, a current crescent forms a candidate set for node $k+1$ with cardinality Z_k that is, conditionally, a Poisson random variable with conditional expected value

$$E[Z_k | L_k = l_k] = S_c(l_k). \tag{2.6}$$

A spoke stops at stage k when the crescent C_k is empty and thus spoke generation is a transient process. The outage constraint depends only on the crescent sizes $S_c(L_k)$ but not on the hop direction process Θ_k. On the other hand, the spoke wobbliness depends on the Θ_k but is meaningful only as long as each current crescent C_k is non-empty. Thus, we separate the analysis of the outage and wobbliness constraints by formally defining $[L_k]$ as a fictitious process that never encounters an empty crescent.

Under the fictitious process model, the position of node $k+1$ will be uniformly distributed over the crescent C_{k+1}. From Figure 2.2 (b) we see that, given the current hop length $L_k = l_k$, the arc of radius ρ has length $2\rho\beta(l_k, \rho)$. The conditional probability that we find node $k+1$ in the annular segment of width $d\rho$ along the arc of radius ρ is $2\rho\beta(l_k, \rho)d\rho/S_c(l_k)$. It follows that the conditional pdf of the next hop length L_{k+1} given $L_k = l_k$ is

$$f_{L_{k+1}|L_k}(\rho|l_k) \quad = \tfrac{2\rho\beta(l_k,\rho)}{S_c(l_k)} \quad R - l_k \leq \rho \leq r, \tag{2.7}$$

and zero otherwise. We note that (2.7) provides a complete characterization of the fictitious process $[L_k]$.

2.2.3 Ergodic Finite State Markov Chain Model

Here, we develop a Markov Chain model that approximates the Markov process described above. We start by quantizing the L_k process, yielding the m-state Markov chain $\hat{L}_k$. We first select a chain state set that quantizes the process state space $[R-r, r]$, then describe a mapping from the process state space to the chain state set and, last, describe the resulting chain probability transition matrix. We define $\{h_1, \ldots, h_m\} \subseteq [R-r, r]$ to be the chain state set. Without loss of generality, we assume that $h_0 = R - r < h_1 < h_2 < \ldots < h_m = r$. As illustrated in Figure 2.3, whenever the kth hop Markov chain state is $\hat{L}_k = h_i$, the corresponding next process hop length is $L_{k+1} \in \mathcal{I}_i = [R - h_i, r]$, where $\mathcal{I}_i$ is the *next hop span* and its length $|\mathcal{I}_i|$ is also the width of the corresponding quantized crescent $\hat{C}_k$ of area $c_i = S_c(h_i)$. L_{k+1} is quantized to state h_j whenever $\hat{L}_{k+1} \in \mathcal{I}_{ij}$ where

$$\mathcal{I}_{ij} \quad = \quad \mathcal{I}_i \cap (h_{j-1}, h_j]. \tag{2.8}$$

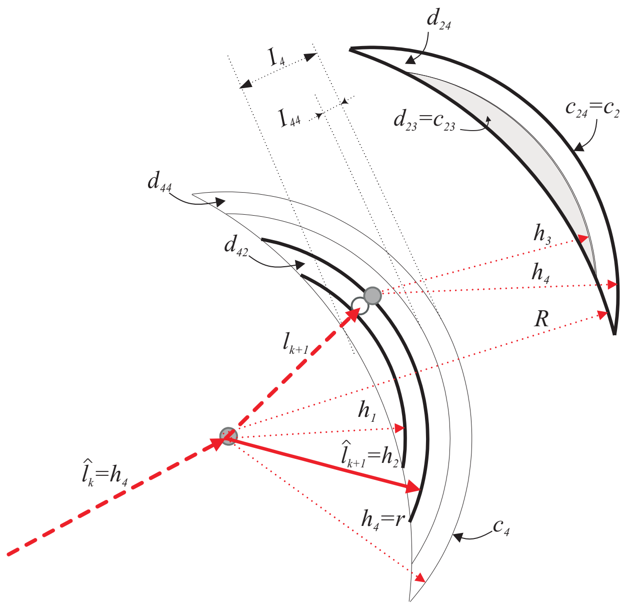


Figure 2.3: Ergodic Finite State Markov Chain: quantization example for a four-state chain ($m = 4$): $\hat{L}_k = h_4 = r$ results in the first crescent $\hat{C}_k$ of area c_4 partitioned into four strips of total area $c_4 = d_{41} + d_{42} + d_{43} + d_{44}$; $L_{k+1} \in \mathcal{I}_{42}$, quantized to $\hat{L}_{k+1} = h_2$, is followed by a crescent $\hat{C}_{k+1}$ of area c_2 and a hop span $I_2 = [R - h_2, r]$ which is (uniformly) quantized into a crescent of area $d_{23} = c_{23}$ (shaded region) and a crescent strip $d_{24} = c_2 - c_{23}$ (the unshaded area).

Note that the set $[\mathcal{I}_{ij} : j = 1, \ldots, m]$ partitions I_i and serves as a set of quantization intervals for L_{k+1} when $\hat{L}_k = h_i$. This quantization mapping is illustrated in Figure 2.3 where $L_{k+1} \in \mathcal{I}_{42}$ is extended to reach the quantized node position marked with a gray circle at $\hat{L}_{k+1} = h_2$. The chain proceeds by declaring a fictitious node at the quantized position as the new leading relay. As depicted in Figure 2.3, a quantization interval $\mathcal{I}_{ij}$ corresponds to the strip of area

$$d_{ij} \quad = \quad \begin{cases} \int_{R-h_i}^{h_j} 2\rho\beta(h_i, \rho)\, d\rho, & j = j^*(i), \\ \int_{h_j-\Delta}^{h_j} 2\rho\beta(h_i, \rho)\, d\rho, & j > j^*(i), \end{cases} \tag{2.9}$$

(and zero otherwise), and of width $|\mathcal{I}_{ij}|$ within the crescent $\hat{C}_k$ of area $c_i = \sum_j d_{ij}$. Here $j^*(i) = \min\{j : h_j > R - h_i\}$ is the index of the leftmost non-empty quantization interval within I_i.

As shown in Figure 2.3, $c_{ij} = S_{\text{IC}}(h_i, h_j)$ is the *quantized interior crescent area* formed by the control circle (of radius R) centered at the kth hop relay and a circle of radius h_j centered at node $k+1$ at distance $\hat{L}_k = h_i$. Note that $c_{ij} < c_{i(j+1)} \cdots < c_{im}$, where $c_{ij} = 0$ for $j < j^*(i)$, $c_{im} = c_i$, and $d_{ij} = c_{ij} - c_{i(j-1)}$. The hop-length transition probabilities

$$\begin{aligned} P_{ij} &= \Pr\{\hat{L}_{k+1} = h_j | \hat{L}_k = h_i\} \\ &= \Pr\{L_{k+1} \in \mathcal{I}_{ij} | L_k = h_i\} = d_{ij}/c_i \end{aligned} \qquad (2.10)$$

follow from the uniformity of Poisson spatial distribution of nodes and since the fictitious process assumes that the crescent $\hat{C}_k$ is not empty. Intuitively, when m is sufficiently large, the ergodic Markov chain will approximate well the ergodic Markov process. Driven by the modeling criteria of simplicity and efficiency, we consider Markov chain models with both uniform and non-uniform quantization of $[R-r, r]$. With only $m = 2$ levels, the uniform quantization lacks accuracy. However, a carefully chosen two-state chain provides a useful non-uniform quantization model. The transition matrix for both two-state systems is

$$\mathbf{P} = \begin{bmatrix} 0 & 1 \\ c_{21}/c_2 & (c_2 - c_{21})/c_2 \end{bmatrix}, \qquad (2.11)$$

since $c_{12} = c_1$ and $c_{22} = c_2$.

Uniform Quantization Model

In this model, the hop-length states $\{h_i\}$ uniformly quantize the process state space $[R-r, r]$ so that $h_i = R - r + i\Delta$, where $\Delta = (2r - R)/m$ is the quantization interval. Furthermore, $j^*(i) = m - i$ so that the next-hop quantization intervals $\mathcal{I}_{ij}$ satisfy $\mathcal{I}_{ij} = (h_j - \Delta, h_j]$ for $j > m - i$ and are empty for $j \leq m - i$. The transition probabilities are now

$$P_{ij} = \frac{c_{ij} - c_{i(j-1)}}{c_i}, \quad i + j > m, \qquad (2.12)$$

and $P_{ij} = 0$ whenever $i + j \leq m$ follows since, in that case, $(h_{j-1}, h_j]$ and $\mathcal{I}_i = [R - h_i, r]$ intersect in at most one point. For example, the uniformly quantized $m = 2$ Markov chain has $\Delta = r - R/2$, $h_1 = R - r + \Delta = R/2$ and $h_2 = r$, and, accordingly, $c_1 = S_c(R/2)$ and

$c_2 = S_c(r)$.

Non-Uniform Quantization Model

Non-uniform quantization, being inherently more complex than uniform, qualifies only if its application renders a simple two-state model possible. The proposed non-uniform quantization, two-state Markov chain model has a simpler definition with $c_1 = 1$, and $c_2 = S_c(r)$. The corresponding set of hop length states includes $h_2 = r$ and h_1, which is a solution of $c_1 = 1 = S_c(h_1)$. Hence, the next hop partition mapping satisfies $d_{21} = c_{21}$, and $d_{22} = c_2 - c_{21}$. Let $R/2 > h_1 = S_c^{-1}(1) > R - r$, and, in this case, we have that $j^*(1) = 2$, $d_{11} = 0$, and $d_{12} = c_1 = 1$. The non-uniform partitioning differs from the uniform in that $c_2 \gg c_1$ and $c_{22} \gg c_{21}$ for large enough r. The rationale behind such a design follows in the next section.

2.2.4 The Spoke Direction Process

Figure 2.2 (a) indicates that the angular hop displacement Φ_{k+1} at hop $k+1$ changes the current spoke direction in that

$$\Theta_{k+1} = \Theta_k + \Phi_{k+1} \quad = \quad \sum_{i=1}^{k+1} \Phi_i. \tag{2.13}$$

We observe that all points along the radius ρ arc in Figure 2.2 (b) are equiprobable locations for node $k+1$. Thus, given the sequence $[L_k]$, the angular hop displacements $[\Phi_k]$ form a sequence of conditionally independent uniform random variables with the conditional pdf

$$f_{\Phi_{k+1}|L_k,L_{k+1}}(\phi|l_k, l_{k+1}) \quad = \quad \frac{1}{2\beta(l_k, l_{k+1})}, \tag{2.14}$$

for $|\phi| \leq \beta(l_k, l_{k+1})$, and zero otherwise. This probability distribution does not change when the conditioning sequence contains quantized values $\{\hat{L}_k\}$. The current angle sequence $\{\Theta_k\}$ is a random walk process modulated by the Markov chain $\{\hat{L}_k\}$, completely described by equations (2.10) and (2.14).

The transform domain analysis of a Markov Modulated Random Walk (*MMRW*) [50] dictates that we first define the conditional moment generating functions of the incremental angular displacement Φ_{k+1} from (2.13)

$$\begin{aligned} g_{ij}(\omega) &= E\left[\exp\left(\Phi_{k+1}\omega\right)|\hat{L}_k = h_i, \hat{L}_{k+1} = h_j\right] \\ &= \frac{1}{2\varphi_{ij}}\int_{-\varphi_{i,j}}^{\varphi_{i,j}} \exp\left(\phi\omega\right)\, d\phi \\ &= \hbar\left(\varphi_{ij}\omega\right), \end{aligned} \tag{2.15}$$

for ω in a convergence region $(\omega_{\text{-}}, \omega_{+})$, where

$$\hbar\left(x\right) = \frac{\sinh x}{x}, \tag{2.16}$$

and

$$\varphi_{ij} = \beta(h_i, h_j). \tag{2.17}$$

We create a matrix $\mathbf{\Gamma}(\omega)$ with elements

$$\Gamma_{ij}\left(\omega\right) \quad = \quad P_{ij}g_{ij}(\omega). \tag{2.18}$$

The Perron-Frobenius theorem (see e.g., [51]) dictates that its largest eigenvalue $\sigma\left(\omega\right)$ is real and positive. The elements of the corresponding right eigenvector $\nu\left(\omega\right) = \left[\nu_1\left(\omega\right)\cdots\nu_m\left(\omega\right)\right]^T$ are also real and positive.

Next, we define the product martingale [50]

$$M_k\left(\omega\right) = \frac{\exp\left(\omega\,\Theta_k\right)\nu_{i(k)}\left(\omega\right)}{\sigma^k\left(\omega\right)\nu_{i(0)}\left(\omega\right)} \tag{2.19}$$

where $i(k)$ is the random state index of the chain at time k, and the random variable $\nu_{i(k)}\left(\omega\right)$ is the $i(k)$-th element of the right eigenvector. This martingale is the key to our analysis of the wobbliness process in 2.4, since it captures its Markov-modulated random walk properties, and allows an elegant application of the random-walk stopping time theory [50].

2.3 Outage Constraint

Here we evaluate the outage probability for given $q = R/r$, in order to evaluate the associated outage constraint (2.22). With respect to outage, a spoke stops at hop k when the crescent C_k is empty, i.e., $Z_k = 0$. Since the nodes obey a planar Poisson process, it follows from (2.6) that the conditional probability the crescent C_k of area $S_c(l_k)$ is empty is

$$\Pr\{Z_k = 0 | L_k = l_k\} = e^{-S_c(l_k)}. \tag{2.20}$$

We define

$$D = \min\{n : Z_n = 0\} \tag{2.21}$$

as the first time the process encounters an empty crescent.

For analytical tractability, instead of requiring the spoke to travel distance d with high probability, we require it to travel η hops with high probability. In particular, we define $\eta = \lceil d/r \rceil$ as the number of hops corresponding to an idealized straight-line spoke extending to the distance d. The design outage constraint can be formalized as

$$\Pr\{D \leq \eta\} \quad \leq \quad p. \tag{2.22}$$

However, the analysis of (2.22) is challenging due to the complex way in which the hop length process $[L_k]$ evolves with time. In particular, a small L_k will create a small crescent; this induces a support set $[R - L_k, r]$ for L_{k+1} that excludes small hop lengths in the interval $[R - r, R - L_k)$. As illustrated in Figure 2.4, an imaginary coil is attached between a fixed pivot and a moving leading relay: when contracted, it pulls the leading relay's data circle inside the blocking control circle, exposing only a tiny area with possible relays. Note that the next hop length has to be long (close to r), if the relay is found in this tiny area. At the other extreme, when the coil is completely relaxed to length r, it exposes the largest possible area. This reduces the likelihood of an empty crescent yet it increases the likelihood of the next hop length being small. This oscillatory effect illustrates the importance of the Markov property for the hop length evolution model (2.7).

For the m-state Markov chain, let us denote the event that the first η crescents $\hat{C}_k$, $k =$

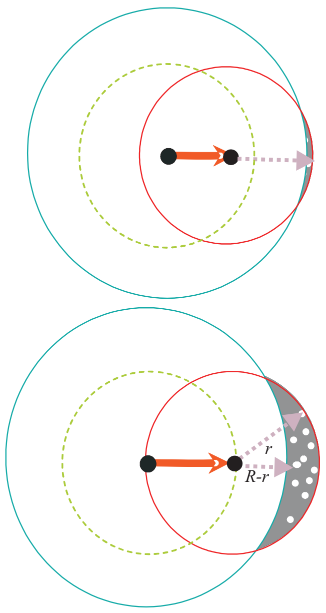


Figure 2.4: Spring-coil analogy

$1, \cdots, \eta$, are not empty as

$$A_\eta = \left\{ \min_{k \le \eta} Z_k > 0 \right\}. \tag{2.23}$$

The probability that the crescents $\hat{C}_1, \ldots, \hat{C}_\eta$ are not empty, and that the system is in state j at time η is denoted

$$\kappa_j^{(\eta)} = \Pr\left\{ \hat{L}_\eta = h_j, A_\eta \right\}. \tag{2.24}$$

Using Markovity of $\hat{L}_k$ and conditional independence of Z_k given $\hat{L}_k$, it is straightforward to show that

$$\kappa_j^{(\eta)} = \sum_{i=1}^{m} \overline{e_j} P_{ij} \kappa_i^{(\eta-1)} \tag{2.25}$$

where $\overline{e_j} = 1 - \exp(-\lambda c_j)$ is the probability of a non-empty crescent while in state j.

Let us define the $m \times m$ matrix $\breve{\mathbf{P}}$ where

$$\breve{P}_{ij} = P_{ij} \overline{e_j} \tag{2.26}$$

is the conditional probability to transition from state i to state j, and that the resulting crescent of area c_j is not empty. Note that (2.11) implies $P_{11} = \breve{P}_{11} = 0$. In addition, by defining the vector $\kappa^{(\eta)} = [\kappa_1^{(\eta)}, \cdots, \kappa_m^{(\eta)}]$, (2.25) becomes

$$\kappa^{(\eta)} = \kappa^{(\eta-1)}\breve{\mathbf{P}}. \tag{2.27}$$

Recursively, we obtain

$$\kappa^{(\eta)} = \kappa^{(1)}\breve{\mathbf{P}}^{\eta-1}. \tag{2.28}$$

Given the initial state m, we see that $\kappa_i^{(1)} = 0$ for $i < m$ and $\kappa_m^{(1)} = \overline{e_m}$. Thus,

$$\kappa^{(\eta)} = [0\ \cdots\ \overline{e_m}]\, \breve{\mathbf{P}}^{\eta-1}. \tag{2.29}$$

As

$$\Pr\{A_\eta\} = \sum_{i=1}^{m} \kappa_i^{(\eta)} = \kappa^{(\eta)}\, [1\ \cdots\ 1]^T, \tag{2.30}$$

the probability that the spoke will stop at or before hop η (assuming that the chain always starts in state h_m) becomes

$$\begin{aligned}\Pr\{D \le \eta\} &= 1 - \Pr\{A_\eta\} \\ &= 1 - [0 \cdots \overline{e_m}]\, \breve{\mathbf{P}}^{(\eta-1)}\, [1 \cdots 1]^T. \end{aligned} \tag{2.31}$$

The following asymptotic (large r) analysis of the outage probability (2.31) is based on the two state non-uniform quantization model (2.11). Let $\lambda_1 > \lambda_2$ be the two eigenvalues of $\breve{\mathbf{P}}$ in (2.31) based on (2.11). The eigenvalue λ_1 describes the rate at which the outage probability increases with the number of hops, while the negative eigenvalue λ_2 describes the oscillatory, self-recovery mechanism depicted in Figure 2.4. Let $r \gg 1$ and q be close to two. Then $c_{22} = c_2 \gg c_1 = 1$ in (2.11). Now, λ_1 is close to one, while λ_2 one is close to zero. Furthermore, by combining (2.31) and (2.22), while expressing the two-state $\breve{\mathbf{P}}$ through its singular value decomposition, and truncating the Taylor expansions of λ_1 and λ_2 to their significant terms, we

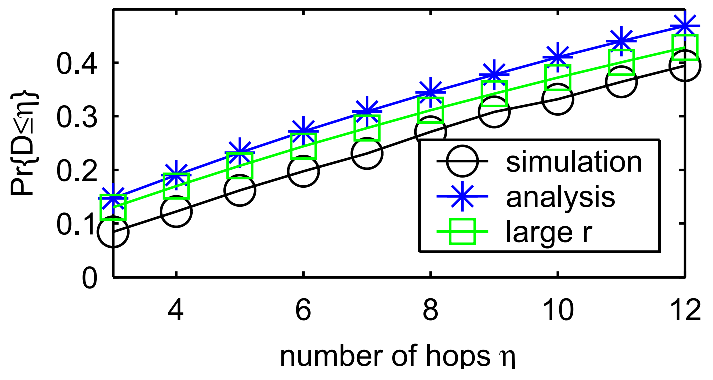


Figure 2.5: Outage probability curves

show that, for a spoke to reach η hops with probability p, given q, the range is required to be

$$r \geq 1/\sqrt{\exp(1)\left(1-(1-p)^{\frac{1}{\eta-1}}\right) f(q)}, \tag{2.32}$$

where $f(q) = S_c(r)/r^2$. Details of derivation of this very important closed-form expression for r are given in the appendix.

Figure 2.5 illustrates how well (2.32) matches the simulation results for a large r. We fix q, and evaluate (2.31) for a sufficiently large r, resulting in the asterix-marked outage probability curve, then plot the simulation statistics for the same pair of q and r (circle-marked curve), and finally, from (2.32), we express the outage probability bound p as a function of η parametrized by the same values of the design parameters (square-marked curve).

2.4 Wobbliness Constraint

The spoke goes off-course at hop k whenever the current angle Θ_k in (2.13) exceeds one of the following two thresholds ϕ_o and $-\phi_o$. To describe spoke wobbliness, we define

$$T_{\varphi_o} = \min\left[k : |\Theta_k| \geq \varphi_o\right]. \tag{2.33}$$

to be the first time that the spoke goes off-course. As we model the angle process evolution only up to that point, T_{φ_o} is the *stopping time* of the random walk Θ_k modulated by the ergodic

Markov chain $\hat{L}_k$. Following [50, Chapter 7.7], T_{φ_o} is also a stopping rule for the martingale $M_k(\omega)$ relative to the joint process $\{M_k(\omega), L_k;\}$

$$M_k(\omega) = \frac{\exp(\omega\Theta_k)\nu_{i(k)}(\omega)}{\sigma(\omega)^k \nu_{i(0)}(\omega)}. \tag{2.34}$$

Hence, following [50, Lemma 6] and the *optional sampling theorem* [50, Theorem 6] we have

$$E\left[M_{T_{\varphi_o}}(\omega)\right] = E\left[\frac{\exp(\omega\Theta_{T_{\varphi_o}})\nu_{i(T_{\varphi_o})}(\omega)}{\sigma(\omega)^{T_{\varphi_o}}\nu_{i(0)}(\omega)}\right] = 1, \tag{2.35}$$

for $\omega \in (\omega_{-}, \omega_{+})$. Since the stopping time T_{φ_o} is a random variable of unknown probability distribution, elaborate mathematical methods must be used to model it. Our methods utilize (2.35), which is an extension of the *Wald identity* to Markov modulated random walks. The first wobbliness constraint is based on the first moment of T_{φ_o}, as

$$E[T_{\varphi_o}] \geq \eta. \tag{2.36}$$

The second wobbliness constraint is based on the *cumulative distribution function (CDF)* of T_{φ_o}, as follows

$$\Pr\{T_{\varphi_o} \leq \eta\} \leq p_t. \tag{2.37}$$

In subsection 2.4.1 we demonstrate how to compute the mean $E[T_{\varphi_o}]$. Subsection 2.4.2 describes a bound on the CDF of the stopping time. These two approaches together provide a good description of the stopping time, based on which a range of q values can be found for each φ_o.

2.4.1 Expected Threshold Crossing Time

The random variable $\Theta_{T_{\varphi_o}}$ is either $-\varphi_o$ or φ_o, assuming that there is no overshoot. We address the problem of overshoot later. By symmetry arguments, first and second moments of $\Theta_{T_{\varphi_o}}$ are

$$\begin{aligned} E\left[\Theta_{T_{\varphi_o}}\right] &= 0, \\ \text{var}\left[\Theta_{T_{\varphi_o}}\right] &= E\left[\Theta^2_{T_{\varphi_o}}\right] = \varphi_o^2. \end{aligned} \tag{2.38}$$

We evaluate the second derivative of (2.35) with respect to ω at $\omega = 0$, and denote

$$\mu_i(\omega) = \nu_i''(\omega)/\nu_i(\omega), \tag{2.39}$$

to obtain the expected number of hops until the hop angle hits the threshold as

$$E\left[T_{\varphi_o}\right] = \frac{\text{var}\left[\Theta_{T_{\varphi_o}}\right] + E\left[\mu_{i(T_{\varphi_o})}(\omega)\right]\Big|_{\omega=0} - \mu_{i(0)}(\omega)\Big|_{\omega=0}}{\frac{\sigma''(\omega)}{\sigma(\omega)}\Big|_{\omega=0}}. \tag{2.40}$$

All the derivations leading to (2.40) are presented in the appendix. We also show that, for $m = 2$, the denominator of (2.40) is

$$\left.\frac{\sigma''(\omega)}{\sigma(\omega)}\right|_{\omega=0} = \frac{1}{3}\left(\pi_2 P_{22}\varphi_{22}^2 + \pi_1 P_{11}\varphi_{11}^2 + \pi_1 P_{12}\varphi_{12}^2 + \pi_2 P_{21}\varphi_{21}^2\right), \tag{2.41}$$

where $\pi_i,\ i = 1, 2$ are the elements of the vector of stationary state probabilities $\overline{\pi} = [\pi_i]_{(1\times m)}$. Note that terms $\varphi_{ij}^2/3$ are transition-specific variances (for uniform angular displacement).

Direct generalization of (2.41) to an m state model has a form of a stationary average of transition-specific variances over m^2 transitions

$$\left.\frac{\sigma''(\omega)}{\sigma(\omega)}\right|_{\omega=0} = \text{var}\,[\theta]_p = \overline{\pi} P^{(v)} u^T,$$

where, $P^{(v)} = \left[P_{ij}^{(v)}\right]_{(m\times m)}$ with elements $P_{ij}^{(v)} = \left(P_{ij}\varphi_{ij}^2\right)/3$, and $u = [1...1]_{(1\times m)}$. Now, it can be shown that, for small crescent subtending angles relative to the threshold φ_o, we can ignore the terms $E\left[\mu_{i(T_{\varphi_o})}\right]$ and $\mu_{i(0)}$ in (2.40), thus

$$E\left[T_{\varphi_o}\right] = \frac{\text{var}\left[\Theta_{T_{\varphi_o}}\right]}{\text{var}\,[\theta]_p}. \tag{2.42}$$

Since (2.40) neglects the overshoot, we now seek to include the overshoot impact. We start with the overshoot analysis of the simple random walk $\Theta_n = \sum_{i=1}^n \Phi_i$, modulated by one-state Markov Chain, i.e. $\Phi_i \sim U(-\varphi_{11}, \varphi_{11})$. Based on the derivation presented in the appendix, which assumes that undershoot and overshoot have the same uniform distribution, we obtain the overshoot-inclusive form of var $\left[\Theta_{T_{\varphi_o}}\right]$ for a *one-state MMRW* model

$$\operatorname{var}\left[\Theta_{T_{\varphi_o}}\right] \;=\; \varphi_o^2 + (2/3)\varphi_o\varphi_{11} + \varphi_{11}^2/6. \tag{2.43}$$

Hence, the overshoot-inclusive form of the numerator of (2.42) contains two additional terms, resulting, for this simple model, in a corrected value of the expected threshold crossing time. We now inductively derive the analytical forms of these terms that are applicable to a m-state Markov-modulated random walk model.

Note in the second term of (2.43) that φ_{11} represents the half-span of the uniform pdf characterizing the angular deviation in each hop. For a m-state Markov-modulated random walk, we replace φ_{11} with a weighted sum of transition-specific angle spans $\sum_{i,j=1}^{n} w_{ij}\varphi_{ij}$, where $w_{ij} = \pi_i P_{ij}$. Hence, the angular deviation span associated with the trivial transition of the one-state MC is now replaced with a stationary average taken over angle-spans associated with m^2 transitions of the m-state MMRW.

Note now that the third term of (2.43) represents one half of the variance for the one-hop angular deviation described by $\Phi_i \sim U\left(-\varphi_{11}, \varphi_{11}\right)$. For the m-state MMRW, we replace this term with another weighted sum where $(w_{ij}/2)$-weighted terms are transition specific variances $\varphi_{ij}^2/3$. Hence, using matrix notation, the extended form of (2.43) becomes

$$\operatorname{var}\left[\Theta_{T_{\varphi_o}}\right] = \varphi_o^2 + 2\varphi_o\left(\overline{\pi}P^{(a)}u^T\right) + 1/2\left(\overline{\pi}P^{(v)}u^T\right), \tag{2.44}$$

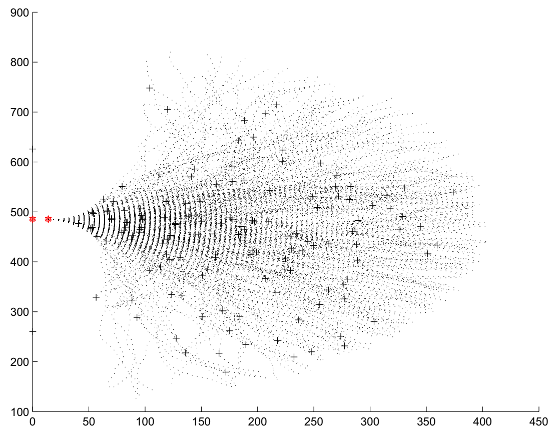
where

$$P^{(a)} = \left[P_{ij}^{(a)}\right]_{(m\times m)} \tag{2.45}$$

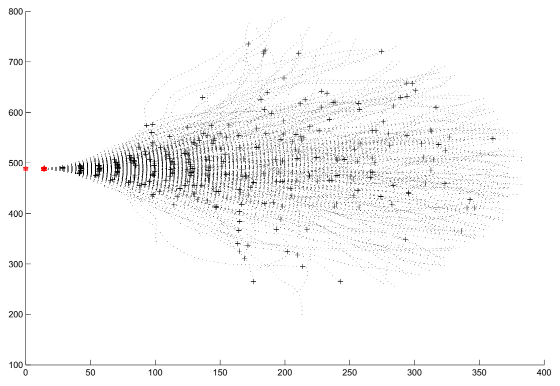
and

$$P_{ij}^{(a)} = P_{ij}\varphi_{ij}/3. \tag{2.46}$$

The overshoot-inclusive variant of (2.42) for a multi-state chain (2.44) yields values that match the simulation results closely and consistently. Figure 3.5 (a) illustrates the achieved wobbliness in a sample of 500 spokes directed eastward, designed to propagate 160 length units with the wobble threshold of $\pi/4$. It is evident that a large number of spokes exceed the targeted propagation distance, while the straightness needs to be improved. Such a behavior



(a)



(b)

Figure 2.6: Sample of spokes directed eastward - ”constraint in the mean” vs.”probability constraint” design.

is due to the fact that the outage constraint is a constraint in probability, while (2.36) is a constraint in the mean, where the pertinent pdf is long-tailed.

2.4.2 Probability of Threshold Crossing Before Time T_{φ_o}

Motivated by the observations illustrated by Figure 3.5 (a), we here analyze the wobbliness model, as defined in (2.37), from the point of view of *Large Deviation Theory (LDT)*. We determine a bound for Pr $\{T_{\varphi_o} \leq \eta\}$ based on the Gärtner-Ellis theorem [51, Thm 2.3.6] and its application to an empirical measure of finite Markov Chains, in particular [51, Exercise 3.1.4].

Let $\wp^{P}_{i(0)}$ denote the Markov probability measure associated with the transition probability matrix (2.10), and with the initial state $\hat{L}_0 = i(0)$. Precisely,

$$\wp^{P}_{i(0)}\left(\hat{L}_1 = y_1, \cdots, \hat{L}_n = y_n\right) = P_{i(0)y_1} \prod_{i=1}^{n-1} P_{y_i y_{i+1}} \tag{2.47}$$

is the probability of a specific Markov chain path, starting at $i(0)$, and transitioning through the sequence of states $[y_i]_{i=1}^{n}$. Now, let us denote $\psi_{ij} = U\left(\varphi_{ij}, \varphi_{ij}\right)$, and thus, the conditional law of $[\Phi_k]$ for each realization $[L_k = y_k]_{k=1}^{n}$ is $\prod_{i=1}^{n} \psi_{y_{k-1}y_k}$.

Denoting with $E^{P}_{i(0)}[.]$ the expected value with respect to $\wp^{P}_{i(0)}$, we further define $\Lambda_n(n\omega) = \log E^{P}_{i(0)}\left[e^{\omega \sum_{k=1}^{n} \Phi_k}\right]$. Following a derivation analogous to [51, Thm 3.1.2], we find that the logarithmic moment generating function of the current angle is related to the largest eigenvalue $\sigma(\omega)$ of (2.18) as

$$\Lambda(\omega) \triangleq \lim_{n\to\infty} \frac{1}{n} \Lambda_n(n\omega) = \log \sigma\left(\omega\right). \tag{2.48}$$

According to [51, Thm 3.1.2], the empirical mean of the sum of angle deviations modulated by $\wp^{P}_{i(0)}$ has a rate function, which is a conjugate function of $\Lambda(\omega)$, i.e the Fenchel-Legendre transform

$$\Lambda^{\star}(x) = \sup_{\omega} \left[\omega x - \Lambda(\omega)\right]. \tag{2.49}$$

A geometric interpretation of $\Lambda^{\star}(x)$ is given in Figure 2.7. Using the fact that $\Lambda(\omega)$ is a convex function, and $\Lambda(\omega) \geq 0$ for $\omega \in (\omega_{-}, \omega_{+})$, and applying the total probability formula over the event space

$$E_1 = \left\{T_{\varphi_o} \leq \eta, \Theta_{T_{\varphi_o}} \geq \varphi_o\right\},$$

$$E_2 = \left\{T_{\varphi_o} \leq \eta, \Theta_{T_{\varphi_o}} \leq -\varphi_o\right\},$$

$$E_3 = \left\{T_{\varphi_o} > \eta, \Theta_{T_{\varphi_o}} \geq \varphi_o\right\},$$

$$E_4 = \left\{T_{\varphi_o} > \eta, \Theta_{T_{\varphi_o}} \leq -\varphi_o\right\}$$

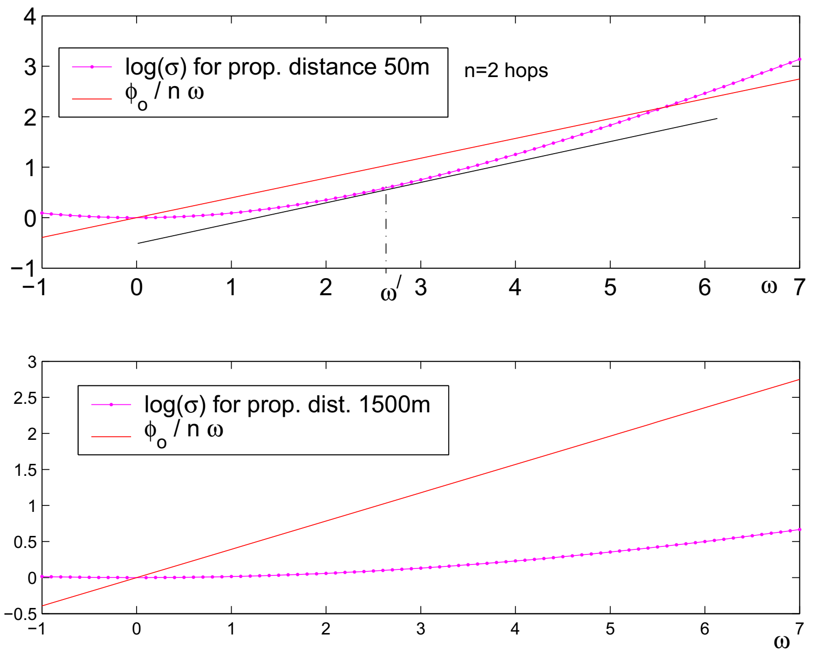


Figure 2.7: **Geometric Interpretation of the bound for Pr $\{T_{\varphi_o} \leq 2\}$ (probability to go off-course in two or less steps)** The **upper** subplot corresponds to a BeSpoken design where the targeted spoke length $d_s \approx 50$m; this design implies a certain maximum subtending angle of the spoke crescents. The **lower** plot corresponds to a design where the desired spoke length $d_l \approx 1500$m, corresponding to smaller maximum subtending angle than the design for d_s - *Consequently, note that the off-course probability after two hops should be smaller for d_l design; the* ***probability bounds*** *relate in the same way: $exp(-2\delta_l) < exp(-2\delta_s)$, where $\delta_s = 0.5$ is the maximum distance (at $\omega = \omega'$ as shown in the plot) between the line $\omega\varphi_o/n$ and $\Lambda(\omega) = \log\sigma(\omega)$, which is exactly (2.49) with $x = \varphi_o/n$, evaluated for d_s design and for $n = 2$. Similarly, $\delta_l > 50$ is the equivalent for d_l design.*

to (2.35), assuming $\omega > 0$, we obtain:

$$\begin{aligned} 1 &= \sum_{k=1}^{4} E\left[\frac{\exp(\omega\Theta_{T_{\varphi_o}})\nu_{i(T_{\varphi_o})}(\omega)}{\sigma^{T_{\varphi_o}}(\omega)\nu_{i(0)}(\omega)}|E_k\right]\Pr\{E_k\} \\ &\geq E\left[\exp(\omega\Theta_{T_{\varphi_o}} - T_{\varphi_o}\log\sigma(\omega))\frac{\nu_{i(T_{\varphi_o})}(\omega)}{\nu_{i(0)}(\omega)}|E_1\right]\Pr\{E_1\} \\ &\geq \exp(\omega\varphi_o - \eta\log\sigma(\omega))\frac{\min_j \nu_j(\omega)}{\nu_{i(0)}(\omega)}\Pr\{E_1\}. \end{aligned} \tag{2.50}$$

Note that $\Pr\{E_1\} = \Pr\{E_2\}$, due to the random walk and the threshold symmetries. Hence,

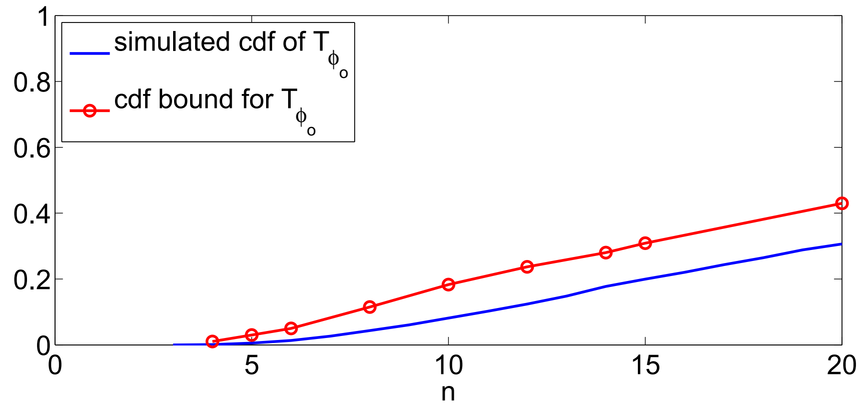


Figure 2.8: Comparison of the LDT-based CDF bound and CDF obtained by ”sampling” the underlying m-state Markov Chain.

by combining the two bounds we have

$$\begin{aligned}\text{Pr}\left\{T_{\varphi_o} \leq \eta\right\} &= \text{Pr}\left\{E_1\right\} + \text{Pr}\left\{E_2\right\} = 2\text{Pr}\left\{E_1\right\} \\ &\leq 2\exp\left(-\eta(\omega\frac{\varphi_o}{\eta} - \log\sigma(\omega))\right)\frac{\nu_{i(0)}\left(\omega\right)}{\min_j \nu_j\left(\omega\right)}, \qquad (2.51)\end{aligned}$$

where $\omega \in (\omega_-, \omega_+)$. We base (2.51) on the largest eigenvalue $\sigma(\omega)$ of an m-state Markov Chain, for sufficiently large m. We apply numerical methods to obtain $\sigma(\omega)$ and observe that (2.51) (with $\nu_{i(0)}\left(\omega\right) / \min_j \nu_j\left(\omega\right) = 1$) bounds the CDF obtained from the simulations, as shown by Figure 2.8. The expression (2.51) evaluated for some desired $\text{Pr}\left\{T_{\varphi_o} \leq \eta\right\} = p_t$ provides an upper bound for q, as opposed to the lower bound obtained through (2.42). Figure 3.5 (b) illustrates the achieved wobbliness in another sample of 500 spokes directed eastward, designed according to (2.51).

2.5 Results and Conclusion

We propose a protocol that generates spokes, relatively straight-line data dissemination trajectories, without requiring the nodes to have navigational information. The BeSpoken protocol implements a simple, spatially recursive process, where a basic set of control packets and a data packet are being exchanged repeatedly among daisy-chained relays that constitute the spoke. Despite the simplicity of the protocol engine, modeling the spoke process is a significant challenge, primarily because it is both a spatial and a temporal random process. The analysis of a Markov-modulated random walk model for the spoke process results in a design conditions which protocol parameters, need to satisfy to produce sufficiently long and straight trajectories. We here summarize the iterative design algorithm for BeSpoken parameters.

```
 Given the desired distance $d$, and the angle threshold $\varphi_o$
$n = 1, \dots \infty$
   (a) Calculate $q^*$ assuming $n = E\left[T_{\varphi_o}\right]$ from (2.42)
   (b) Given $q = q^*$ from (a), calculate $r^*$ from (2.32)
   (c) If $d/r^* < n$, goto (a) else BREAK.
```

Note that (2.42) expresses the expected stopping time as a function of q only. We support our analysis with simulation results. We simulate a stationary network of unit-density, with uniformly distributed nodes deployed over a square region. Simulation statistics are generated by extending a large number of spokes to follow the same direction (as in Figure 3.5), over several network realizations. The collected averages for the spokes going off-course confirm the validity of the design with respect to the imposed constraint ”in the mean” (2.42). By imposing the second wobbliness constraint (2.51), we obtain statistics that show a better control of the spoke direction at the expense of a slightly increased rate of prematurely stopped spokes due to outage (Figure 3.5 (b)). We have also evaluated BeSpoken in its capacity of a spoke-infrastructure building tool, by performing extensive simulations in which several equally spaced spokes are spawned radially from the source. To create this wheel-like pattern of source spokes, we use BeSpoken bootstrap mechanisms, also based on an intersection of two different transmission ranges. In addition, ”randomly placed events” initiate creation of sink-spokes, thus tessellating the simulated plane . An example of such an experiment is given by Figure 1.1, hence putting a face on the proposed design, and illustrating its potential for network applications, other than data dissemination.

Chapter 3

Adaptive BeSpoken: A Step Toward Real Networks

In sensor network deployments, spatial distributions of sensors are frequently far from being uniform [52]. Such networks often contain regions without enough sensor nodes, which we refer to as holes. In routing, holes are communication voids that cause greedy forwarding [25,26] to fail, and the basic BeSpoken protocol, described in Chapter 2, suffers from the same vulnerability. The BeSpoken design parameters were optimized for network nodes scattered as a planar Poisson point process of known uniform intensity. It was essential to match the protocol parameters to the density of network nodes. Whenever the node distribution in the network differs from the assumed Poisson point process, the BeSpoken performance fails to satisfy the design constraints with high probability. We propose to extend the basic BeSpoken protocol with adaptive mechanisms to alleviate problems with small-scale network discontinuities (network thinning caused by random node dying, holes, voids). In this chapter, an adaptive BeSpoken protocol is proposed and its model is analyzed in order to establish a quantitative measure of the protocol performance. Its improvement over the non-adaptive version is evaluated in terms of the gain in the likelihood to achieve a given propagation distance when employing the same protocol parameters.

3.1 BeSpoken Backward Repair Mechanisms

In the following, the crescent to which the relay belongs is *the own crescent*, and the crescent which is formed as a result of relay's transmission is referred to as *the induced crescent*. The *Current Leading Relay (CLR)* activates the BeSpoken adaptive mechanism, since it has the ability to detect an empty candidate set by observing the absence of the RTS request within a time-out period. Upon encountering an empty crescent $S_c(l_k)$ without candidate relays, CLR node k can solicit another pivot from a previous crescent, while keeping its leading-relay status,

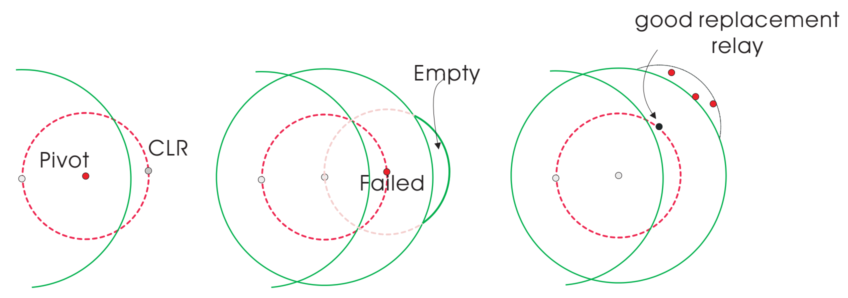


Figure 3.1: The triptych represents a single transition of the Markov process modeling the adaptive spoke: in the *first* step a new relay is selected by the pivot that becomes the Current Leading Relay (CLR); in the *second* step the CLR becomes aware that its induced crescent is empty, when it does not receive any RTS in due time; in the *third* step the CLR solicits its own replacement by sending the request intended for all the nodes in its own crescent - we show one such peer node that replaces the CLR and, having a non-empty induced crescent, repairs the spoke.

which would effectively change the current spoke angle. Another *backward-repair* technique requires the CRL to solicit its own replacement from its own crescent $S_c(l_{k-1})$, i.e., among the peer candidate relays. The latter approach is termed the *one-step backward repair* protocol. The example shown in Figure 3.1 illustrates one possible scenario for the recovery attempt. Here, a replacement relay has been found that creates a large non-empty crescent, hence repairing the spoke. We can envision that even if another replacement relay was selected with an induced crescent smaller than the failed one, but at a sufficient distance from the empty crescent to result in a large enough disjoint area, the chances of repairing the spoke are worth performing this step back.

There is a whole spectrum of alike adaptive algorithms, nevertheless, for the sake of simplicity and without loss of generality, we only consider the one-step backward repair protocol.

3.2 One-Step Backward Repair Model

We here analyze the adaptive mechanism introduced in 3.1, where the current leading relay (kth node) runs into an empty set of relay candidate nodes, and solicits a single own replacement from its own crescent. The spoke stops when the replacement results in a non-empty relay candidate set. Extensions to the cases with multiple replacement trials are straightforward.

To model the adaptive mechanism, we use the uniform quantization model illustrated in Figure 2.3, with only two quantization levels, again, for simplicity and without loss of generalization. Hence, the BeSpoken hop-length evolution is modeled by a two-state Markov Chain $\hat{L}_k$. Following the notation from subsection 2.2.3 , the states 1 and 2 correspond to quantized hop lengths $h_1 = R - r + \Delta = R/2$ and $h_2 = r$, where $\Delta = r - R/2$. The corresponding quantized areas are $c_1 = S_c(R/2)$ and $c_2 = S_c(r)$. To make a better distinction between crescents pertaining to different states, in the present work we will use the notation $s_L = S_c(h_2)$ and $s_S = S_c(h_1)$, to denote the large and the small crescent areas, respectively.

To establish a unifying model for both the hops that are completed without utilizing the adaptive mechanism, and those that are completed in the second attempt, through the adaptive algorithm, we redefine the conditions under which the underlying Markov Chain transitions to a new state. Now the chain does not transition into another state always when the next relay is found, but, as an additional condition, the induced crescent of the selected relay must not be empty. The new definition incorporates a lookahead element with each "hop-length" state to make sure that at least one subsequent hop is possible. This is necessary to seamlessly integrate the adaptive mechanism into the two-state model. This effectively adds a third trapping state T when the subsequent hop is not possible. The transition probabilities that (partially) characterize this redefined Markov Chain form the reduced matrix

$$\mathbf{P_{ad}} = \begin{bmatrix} 0 & P_{12}^A \\ P_{21}^A & P_{22}^A \end{bmatrix}, \tag{3.1}$$

since, as described in (2.11), the chain can not transition to the same state from the state corresponding to the small crescent. The reduced matrix $\mathbf{P_{ad}}$ does not include the transitions to trapping state T. In order to formally define transition probabilities P_{ij}^A, we require additional notation. The data propagation will stop if both the induced crescent of the CLR and of its replacement are empty. Trivially, if there are no replacements in the current crescent, it is sufficient that the first induced crescent is empty. Let us denote the cardinality of the union of the two candidate relay sets (in these two induced crescents) with Z_k^2. Now, it follows that the transition probability (with lookahead) can be expressed as

$$P_{ij}^{A} \quad = \quad \Pr\left\{L_k = h_j, Z_k^2 > 0 | L_{k-1} = h_i, Z_{k-1} > 0\right\}. \tag{3.2}$$

Note that $\mathbf{P_{ad}}$ is the adaptive counterpart (for the *One-Step Backward Repair BeSpoken Protocol*) of the transition matrix $\breve{\mathbf{P}}$ whose elements are given in (2.26).

The envelope of all possible replacing crescents created from state $p = 1$ is shown in Figure C.1. The envelopes for the large current crescent are presented in Figures C.2 and C.3. Figure C.2 fixes the replacement relay's state to state 1, and illustrates two general cases of the relative position of the failed CLR and its replacement. Their relative position determines the intersection of the induced crescents, where the support set of the replacement crescent is approximated with so-called small envelope. Figure C.3 fixes the replacement relay's state to state 2, which brings forth so-called large envelope. In the following we evaluate $\mathbf{P_{ad}}$ for a Poisson node distribution and using area linearization when approximations are necessary.

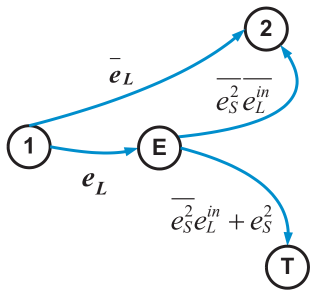
The probability of repair for the two-state Markov chain model, given that the failed leading relay was in state $n \in [1, 2]$, and that the pivot was in the state $p \in [1, 2]$, depends on the position of the replacement relay, more precisely, on its quantization level, and on how much of its induced crescent area is disjoint from the crescent formed by the failed relay. We refer to this induced disjoint areas as the *innovation area*. The average innovation area, denoted with $\overline{S_{in}^{p,n}}$, is calculated as the average difference

$$\Delta^{n,m} = \left(S_c^n \bigcup S_c^m\right) - S_c^n,$$

between the first (failed) crescent area $S_c^n(k+1)$ and the crescent area $S_c^m(k+1)$ formed with the replacement relay. Here, m denotes the quantization state of the replacement relay.Hence,

$$\overline{S_{in}^{p,n}} = \sum_m P_{pm} E\left[\Delta^{n,m}\right]_{\Xi(pm)},$$

where the average is taken first over the envelope of possible crescents induced by replacement relays in state m, denoted with $\Xi\,(pm)$.



$\overline{e_L}$
$\overline{e_S^2}\,\overline{e_L^{in}}$
e_L
$\overline{e_S^2} e_L^{in} + e_S^2$
1
2
E
T

Figure 3.2: Transitions of the adaptive mechanism contributing to the transition probability P_{12}^A; here E denotes the empty crescent in state 2, and T denotes the trapping state, when the repair attempt fails.

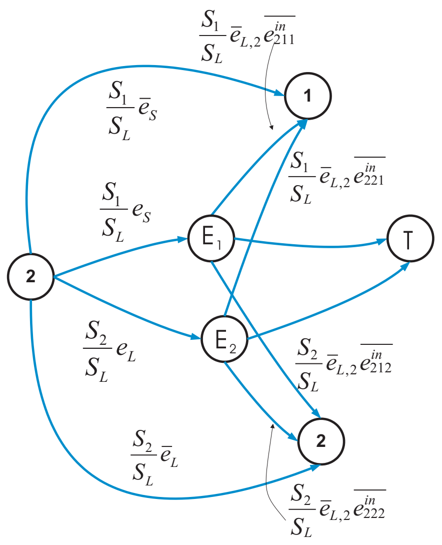
The average probability of finding a non-empty set of candidate relays for the next hop is

$$\overline{e_{p,m,n}^{in}} = E\left[\left(1 - e^{-S_{in}^{m,n}(k+1)}\right)\right]_{\Xi(pm)},$$

given that the pivot was at the state p, the failed relay was at the state n, and that a replacement relay was found corresponding to the MC transition from state p to some state m. Here, the averaging was done over the possible relative positions of the crescents (of respective areas) $S_c^m(k+1)$ and $S_c^n(k+1)$, the relative position being a random variable whose support set is determined by n and p, and ultimately by $\Xi\left(pm\right)$. In the appendix we calculate an approximation for $\overline{e_{p,m,n}^{in}}$ by linearizing both the envelope area and the area of the crescent.

Let $e_L = e^{-s_L}$ and $e_S = e^{-s_S}$ denote the probabilities of large and small crescent being empty, respectively; $\overline{e_L} = (1 - e^{-s_L})$ and $\overline{e_S} = (1 - e^{-s_S})$ denote the probabilities of large and small crescent having at least one node; $\overline{e_{L,2}} = (1 - e^{-s_L}\left(1 + s_L\right))$ and $\overline{e_{S,2}} = \left(1 - e^{-s_S}\left(1 + s_S\right)\right)$, denote the probabilities that at least two nodes will be found within the crescent of area s_L and s_S.

The introduced notation is used to represent probabilities of different repair paths contributing to a particular adaptive BeSpoken transition, as illustrated in Figures 3.2 and 3.3, when pivot is in state 1 and 2, respectively. For example, Figure 3.2 states that to transition from state 1 to state 2 one can transition directly if CLR's induced crescent is not empty. If it is empty, its replacement exists, and its replacement's induced crescent is not empty, it will transition to



$\frac{S_1}{S_L}\bar{e}_{L,2}\overline{e_{211}^{in}}$
$\frac{S_1}{S_L}\bar{e}_S$
$\frac{S_1}{S_L}\bar{e}_{L,2}\overline{e_{221}^{in}}$
$\frac{S_1}{S_L}e_S$
$\frac{S_2}{S_L}e_L$
$\frac{S_2}{S_L}\bar{e}_{L,2}\overline{e_{212}^{in}}$
$\frac{S_2}{S_L}\bar{e}_L$
$\frac{S_2}{S_L}\bar{e}_{L,2}\overline{e_{222}^{in}}$
1
2
E1
E2
T
2

Figure 3.3: Transitions of the adaptive mechanism contributing to the transition probabilities P^A_{21} and P^A_{22}; here E_1 denotes the empty crescent in state 1, E_2 denotes the empty crescent in state 2, and T denotes the trapping state, when the repair attempt fails.

state 2 in the second attempt. Otherwise, it will transition to state T and the spoke will stop. Similarly for Figure 3.3 which describes the repair paths for transitions from state 2. The edges of state diagrams are denoted with probabilities for each corresponding transition event. Hence, by summing the probabilities of the contributing transition repair paths, we obtain the elements P^A_{ij} of the matrix $\mathbf{P_{ad}}$ as follows:

$$P^A_{11} = 0 \tag{3.3}$$

$$P^A_{12} = \overline{e_L} + e_L\overline{e_{S,2}}\overline{e^{in}_{122}} \tag{3.4}$$

$$P^A_{21} = \frac{s_1}{s_L}\left[\overline{e_S} + \overline{e_{L,2}}\left(\frac{s_1}{s_L}e_S\overline{e^{in}_{211}} + \frac{s_2}{s_L}e_L\overline{e^{in}_{221}}\right)\right] \tag{3.5}$$

$$P^A_{22} = \frac{s_2}{s_L}\left[\overline{e_L} + \overline{e_{L,2}}\left(\frac{s_1}{s_L}e_S\overline{e^{in}_{212}} + \frac{s_2}{s_L}e_L\overline{e^{in}_{222}}\right)\right]. \tag{3.6}$$

The derivation details are given in the appendix.

3.2.1 Adaptive Outage Constraint

In order to formalize the Outage constraint for the One-Step Backward Repair protocol, we now define

$$D^A = \min\left\{n : Z_n^2 = 0\right\} \tag{3.7}$$

as the first time the process fails to repair the spoke if it encounters an empty crescent. Hence, the adaptive outage constraint can be expressed as

$$\Pr\left\{D^A \le \eta\right\} \quad \le \quad p. \tag{3.8}$$

Let us denote the event that the spoke does not stop in the first η hops as

$$A_\eta^A = \left\{\min_{k\le\eta} Z_k^2 > 0\right\}.$$

The probability that the spoke does not stop in the first η hops, and that the system is in state j at time η is

$$\mu_j^{(\eta)} = \Pr\left\{\hat{L}_\eta = h_j, A_\eta^A\right\}.$$

Using Markovity of $\hat{L}_k$ and conditional independence of Z_k^2 given $\hat{L}_{k-1}$, it is straightforward to show that for the adaptive BeSpoken the expression (2.25) becomes

$$\mu_j^{(\eta)} \quad = \quad \sum_{i=1}^{m} P_{ij}^A \mu_i^{(\eta-1)}. \tag{3.9}$$

By defining the vector $\mu^{(\eta)} = [\mu_1^{(\eta)}, \cdots, \mu_m^{(\eta)}]$, (3.9) becomes $\mu^{(\eta)} = \mu^{(\eta-1)}\mathbf{P_{ad}}$. Recursively, we obtain $\mu^{(\eta)} = \mu^{(1)}\left(\mathbf{P_{ad}}\right)^{\eta-1}$. Given the initial state m, and $\mu_i^{(1)} = 0$ for $i < m$, $\mu_m^{(1)} = \overline{e_m}$, we obtain $\mu^{(\eta)} = [0\ \cdots\ \overline{e_m}]\left(\mathbf{P_{ad}}\right)^{\eta-1}$.

As $\Pr\left\{A_\eta^A\right\} = \sum_{i=1,\cdots,m}\mu_i^{(\eta)} = \mu^{(\eta)}\left[1\ \cdots\ 1\right]^T$, the probability that the spoke will stop at or before hop η (assuming that the chain always starts in state h_m) becomes

$$\begin{aligned}\Pr\left\{D^A \le \eta\right\} &= 1 - \Pr\left\{A_\eta^A\right\} \\ &= 1 - [0\cdots\overline{e_m}]\left(\mathbf{P_{ad}}\right)^{(\eta-1)}[1\cdots1]^T.\end{aligned} \tag{3.10}$$

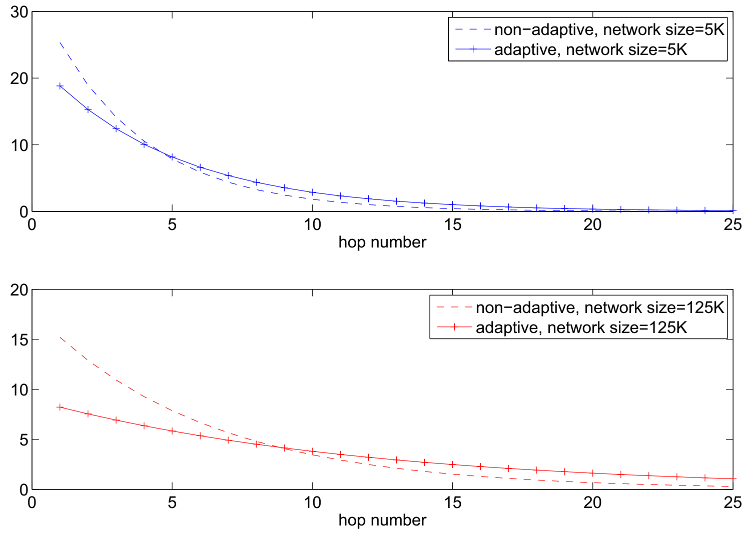


Figure 3.4: Percentage of spokes dying at each hop based on two-state uniformly quantized Markov Chain model: adaptive mechanism decreases the probability of spokes dying prematurely

3.3 Conclusion: Numerical Analysis and Simulation Results

We have numerically evaluated the outage probability based on the two-state uniformly quantized Markov Chain model, both for the basic BeSpoken (2.31) and the presented adaptive version (3.10). We are aware that the small number of quantization levels introduces an error, but the evaluation is comparative, and we expect that the error is unbiased. In both cases Markov Chain transition probabilities were calculated using the same design parameters (based on (2.32) and 2.5). The results presented in Figure 3.4 for two different network sizes show that the adaptive algorithm works better. We also support our analysis with simulation results. For the same random network instances, we ran simulations of basic and adaptive versions of the protocol whose design parameters are based on the guidelines in [53] ((2.32) and the algorithm in 2.5), given a network of uniformly distributed nodes with unit density and known size.

The first experiment was designed to compare the performance of the two protocols for a thinned network whose node density is lower than the one used for the protocol parameter (r and q) design. Hence, the network nodes were deployed in a uniform manner over a square region ensuring a half-unit density. Spoke traces are generated as by extending a large number

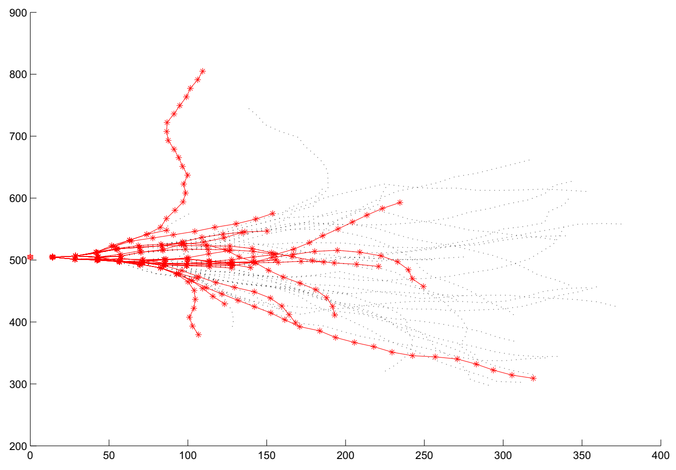


Figure 3.5: Two samples of spokes directed eastward in a *thinning* network where the density of nodes is decreased to one half of the initial density due to random node dying: thin-line spokes are created by the adaptive BeSpoken and asterix-marked spokes created by the basic BeSpoken; One-Step Backward adaptive protocol demonstrates better performance than the basic BeSpoken i.e. more spokes survive in the thin-line cloud that in the cloud of asterix-marked spokes.

of spokes to follow the same direction. The adaptive mechanism used here is the analyzed One-Step Backward adaptive protocol. The overlapping traces of both protocol variants are shown in Figure 3.5. The presented result clearly illustrates that the adaptive BeSpoken performs better in cases where the applied protocol parameters have been underdesigned for the current network density. This is an important observation as any WSN can become scarcely populated due to random nodes dying.

The second simulation experiment was designed to evaluate the expected better performance of the adaptive BeSpoken for a network with a hole (small unpopulated network area). For established design parameters (r and q), we simulated a stationary network of unit density, with uniformly distributed nodes deployed over a square region, where all nodes in the bounded region highlighted in Figure 3.6, have been removed. Again, spoke traces are generated as we extended a large number of spokes to follow the same direction. The overlapping traces of both protocol variants are shown in Figure 3.6. The adaptive mechanism used here is Two-Step Backward adaptive protocol, as we expected that once the hole is encountered, the

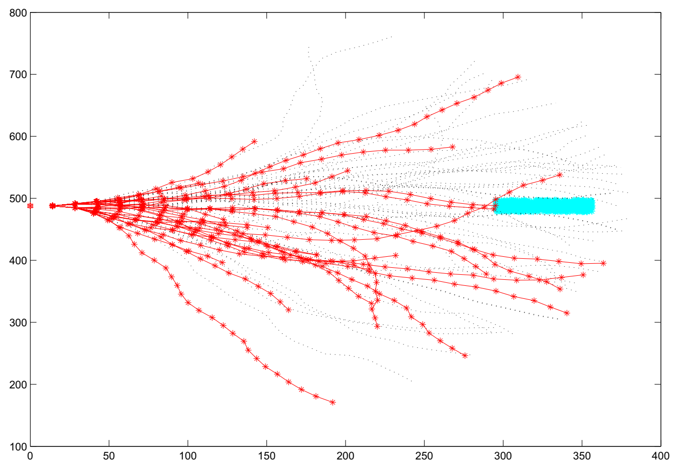


Figure 3.6: Two samples of spokes directed eastward in a network where a *hole* has been created due to node destruction: thin-line spokes are created by the adaptive BeSpoken and asterix-marked spokes created by the basic BeSpoken; Two-Step Backward adaptive protocol demonstrates better performance than the basic BeSpoken.

spoke needs to significantly change its direction in order to avoid the void. The presented result illustrates that this adaptive BeSpoken performs better in the presence of holes. We suggest that each WSN application can be associated with higher probability of irregularities of a particular type. Hence, the appropriate adaptive protocol can be selected according to the application. For example, a WSN deployed for environmental monitoring over long time periods is likelier to suffer from network thinning, while a WSN deployed in disaster areas has higher chances to experience network holes as sensors can be systematically destroyed by hazardous events.

Chapter 4

Data Collection in WSNs Based on Distributed Networked Storage: an Overview

Thematically, the following chapters represent the second part of the dissertation. This chapter provides an overview of the existing work and key problems in WSN distributed networked storage while, practically, serving as an introduction to the second part of the dissertation.

In order to motivate our approach to the problem, in this chapter, we first characterize general coding approaches for distributed networked storage, and, next, describe in detail and quantify the complexity and performance of some widespread distributed coding schemes employed for data dissemination and storage. Note here that we use the term *distributed networked storage*, to distinguish between this paradigm of storing data from multiple sources across multiple storage destinations (within a network) and the framework commonly referred to as *distributed storage* where single source data is stored across multiple (distributed) disks, as suggested in [54]. Finally, we provide a critical discussion of the existing approaches, and motivate the alternative solutions that we propose in the following chapters.

4.1 Encoding Schemes for Networked Storage in WSNs

4.1.1 Connection to Data Persistent Storage

Several papers have recently appeared that analyze distributed coding for data persistent storage in WSNs [35–40]. This body of work is closest to our proposed approach in that the encoding methods are the same and, hence, also provide data ”at the fingertips” of a mobile data collector and, in addition, ensure robustness against failures by storing encoded pieces at a large number of nodes.

A general description of the problem of coding for data persistence in WSNs [38] can be given using the already introduced system model of a WSN with n nodes and k sources at any given moment. Considering that all sensors are inherently unreliable and vulnerable to failures, and, also, compelled to colaborate by relaying each other's data, the question of data persistence naturally arises: how do we complement these features in order to reliably store (and retrieve) all k independent data packets that the sensors have gathered, even after a subset of sensors has failed.

Note that data persistence in WSNs has a spatio-temporal character since it implies some changes over time, and it concerns the whole network area. In addition, it has a random character by virtue of its unreliable storage. Both properties are also associated with the data collection scenario we study here, as the data collector appears at a random position, at random time, and aims to collect all the k source data packets. The common solution to this problem and the data persistence problem is the storage redundancy. Regardless of where the data collector appears, redundant storage should allow for efficient data collection from a compact collection area *at its fingertips*, i.e., at a set of connected (through multiple hops) wireless nodes in its proximity. Redundant data storage across the network may mean simply storing source packet replicas, or random linear combinations thereof, and resulting respectively in a repetition code, or other linear code, implemented across a network.

In conclusion, storing encoded pieces at a large number of nodes provides the diversity of available information needed to achieve the robustness against failures. This diversity is also needed because the location of a potential data collector is unpredictable, and the goal is to make the data available in its proximity. In both applications, such storage is referred to as data persistent storage.

4.1.2 Coding Models

We now address recently introduced random encoding schemes used for storage over wireless sensor networks. The two general classes of packet combining (coding) techniques discussed there are: Fountain-type decentralized erasure codes [37–40, 55], and decentralized erasure codes [36] - a variant of random linear network codes [56, 57], and a distributed version of Maximum Distance Separable codes [58]. Note that both code classes are linear, where the linear combining of source packets is performed over a finite field $GF(q)$. Both classes can be

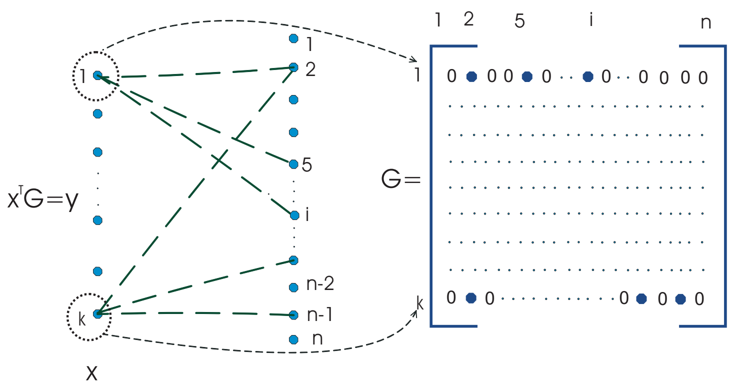


Figure 4.1: Copies of each source packet are stored at $O(\ln n)$ random nodes out of n network nodes; the figure shows how the sets of nodes that sources 1 and k (circled nodes on the left) select to store copies of their packets correspond to non-zero entries (dark dots) in the corresponding rows (1st and kth) of the generator matrix G; non-zero entries are random coefficients from field $GF(q)$. X is the vector of distinct k packets, and Y denotes the vector of n code symbols.

represented by a bipartite coding graph whose vertices correspond to k sources and n storage nodes, as shown in Figure 4.1. The mapping from k source packets to n encoded packets can be also represented by a random generator matrix G whose elements are coefficients randomly selected by storage nodes from $GF(q)$, also illustrated in Figure 4.1. Here, $Y = GX$, X being the vector $[x_1, \cdots, x_k]^T$ of source packets, and Y, the vector $[y_1, \cdots, y_n]^T$ of encoded packets. Precisely, we view the k packets as elements in $GF^s(q)$, i.e., vectors composed of s chunks in a field of size q. Thus, if we denote the chunks by c_j, $j = 1, 2, \cdots k$, then each storage node stores random linear combinations of the c_js. More specifically, if the linear combination stored with a node i is denoted f_i, then

$$\begin{aligned} f_i &= \sum_{j=1}^{k} c_j \beta_j \\ &\text{where} \\ &\Pr\{\beta_j = \beta\} = \frac{1}{q}, \qquad \beta \in GF(q). \end{aligned} \tag{4.1}$$

4.1.3 Connections to Random Network Coding

Traditionally, the information flow in networks has been modeled as multi-commodity flow, where the underlying network consists of error-free links of finite capacity, and where the only operation that can be performed on data is routing (i.e. storing and forwarding packets without modifying their contents). By Mengers theorem, the maximum information that can flow is upper bounded by the value of the minimum cut between the source and the destination; this well-known result from classical graph theory is generalized by the Max-flow Min-Cut theorem [59, 60].

When nodes are allowed to send messages created by combining two or more incoming messages (making linear combinations of packets over some finite field) the situation is much less clear, although the rate dependancy on the network graph is well understood for wired networks. There are examples of directed graphs where a higher communication rate is achievable through these *network coding solutions* than by routing, some of them presented in the seminal paper by Ahlswede et al. [61].

Most of the work to date on network coding focuses on multicast. Now, an equivalent way of thinking of the data collection problem is that of expanding the random bipartite graph connecting the k data nodes with the n storage nodes by adding a data collector for every possible subset of size k of the n storage nodes. Then the problem of *multicasting* the k data packets to all the $\binom{n}{k}$ data collectors is equivalent to making sure that every collection of k storage nodes can reconstruct the original packets. This connection of storage and multicasting was proposed in [36], and reiterated in [54]. It is known that random linear network codes [62] are sufficient for multicasting problems as long as the underlying network can support the required throughput. Decentralized erasure codes can therefore be seen as random linear network codes [62] on the (random) bipartite graph connecting the data and the storage nodes, where each edge corresponds to one routed packet. Note that the communication graph in distributed storage does not correspond to physical links but to virtual routing paths that are made by the randomized algorithm. Therefore, we can design rules for selecting some of the graphs from such a large ensemble of graphs, in order to minimize communication cost.

As already mentioned, several papers have recently offered solutions for data storage in

WSNs, based on various forms of random network coding [35–40]. Except for [37], where the authors provide an interesting analysis of an algorithm for random sampling of WSN nodes, based on random walks with traps, but consider only location-aware network nodes arranged in a grid topology, we here present detailed description of these approaches.

We next outline several important common characteristics of these distributed storage techniques, which are later used as criteria in comparative evaluation of solutions exhibited by each particular approach.

4.1.4 Hallmarks of Random Codes in Distributed Networked Storage

Decentralized Encoding: The information is sensed in multiple locations and global coordination is unavailable. Hence, the code construction should be distributed and based on local knowledge. In particular, each data source chooses where to route its packet independently, and furthermore the storage nodes select their coefficients independently. Algebraically, this corresponds to having a code where every row of the generator matrix G is created independently (as illustrated in Figure 4.1). A code with this row independence property is called *decentralized* [36]. This property calls for stateless randomized network algorithms to generate the encoded information. Note that, in addition, Fountain type decentralized erasure codes require the number of nonzero elements of the matrix columns (or code symbol degree) to have desired statistics. This results in the most appealing feature of Fountain type coding - the linear complexity of decoding which, here, corresponds to linear original data reconstruction time.

Maximum Distance Separable property: Ideally, the data collector would like to reconstruct all k source packets from linear combinations gathered from any set of k storage nodes. The collector must invert a $k \times k$ submatrix $G^{'}$ of G (as in Figure 4.2), corresponding to a set of k nodes. The key property required for successful decoding is that any selection of k nodes (sub-matrix of G) forms a full rank matrix. When this is the case, decoding corresponds to solving a system of linear equations (using for example, Gaussian elimination) over a finite field the random coefficients have been selected from. A matrix that has that property corresponds to a Maximum Distance Separable *(MDS)* code.

Let us point out that such combinatorial constructions are quite difficult to achieve in a distributed manner.

Number of Packet Replicas - Sparcity of code's generator matrix: Both random linear codes and MDS codes have dense generator matrices, which is not desirable for codes constructed in a distributed manner, such as decentralized erasure codes, as significant communication is required to construct such non-sparse matrices. Algebraically, the question is how sparse can a matrix with independent rows be made, and still have the property that square sub-matrices are full rank with high probability. Practically, in most distributed implementations, this question is about how many packet replicas should each souce send to storage nodes.

Network Algorithms for random sampling: All these methods rely on a packet routing layer that can route packets to uniformly random locations in the WSN. This is sometimes referred to as *pre-routing* [36]. Constructing such random sampling algorithms which are distributed and localized is key for the construction of codes in networks. For Fountain coding based storage, the key difficulty is in devising efficient techniques to disseminate data from multiple sources to network storage nodes in a manner which ensures that the required statistics of created linear combinations is accomplished. Achieving this goal is particularly difficult when employed with the classic random geometric graph network models [38–40].

4.2 Existing Solutions to Coding for Persistent Data Storage in WSNs

4.2.1 How many replicas should a source send away?

The following toy example provides more insight into the above stated question. It is based on the distributed version of the coupon collector model [63] (uniform sampling with replacement). The Coupon Collector Problem concerns a shopper who tries, in several attempts, to collect a complete set of k different coupons. Each attempt provides the collector with a coupon randomly chosen from k known kinds, and there is an unlimited supply of coupons of each

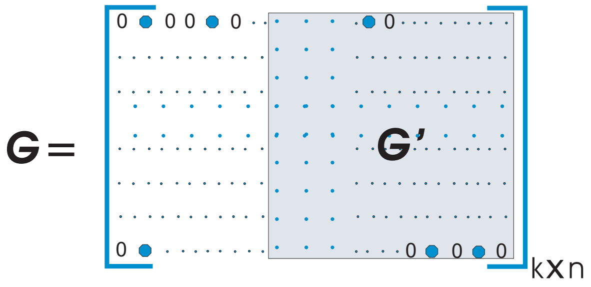


Figure 4.2: Relaxation of MDS requirement due to random (and distributed) construction results in requiring a generator matrix G whose all square submatrices are full-rank *with high probability*. Hence, the random square submatrix $G^{'}$ illustrated in this figure, representing a random collection of storage nodes (and their linear combinations), is expected to be full-rank, thus promising full data recovery from this random set of nodes.

kind. The expected time $E[T_k]$ to collect all k coupons is

$$E[T_k] = 1 + \frac{k}{k-1} + \frac{k}{k-2} + \cdots + k = k(\ln k + O(1/k)).$$

Imagine now a centralized entity that has an access to a repository of packets produced by k sources. Its goal is to draw k different packets from the repository and to store them at k different nodes. There is infinite number of copies of each packet type. So, the centralized entity is expected to access the repository $O(k \ln k)$ times in order to accomplish the goal. This gives us an insight on how many copies of a packet ($O(\ln k)$) each source should distribute independently across the network for sets of k storage nodes to be able to collect enough independent linear combinations of their packets for decoding.

We now explain in detail the variations of this basic approach. In the scheme proposed in [36], the authors consider a large-scale wireless sensor network where the ratio of k and n is held constant as the network grows. They introduce a class of erasure codes mechanized by independent data dissemination from each source node to a randomly selected subset of storage nodes, storage node being any other node in the network. Linear packet combinations stored at each node are determined by random *pre-routing*. Specifically, following an extension of the coupon collector problem, the authors suggest that the data should be diffused by pre-routing $O(\ln n)$ replicas of one source packet node to randomly selected storage nodes. The requirement that $O(\ln n)$ packets be randomly prerouted is based on the requirement that the decoding of the k packets can be performed based on the linear combinations stored at k randomly selected storage nodes in a manner akin to MDS codes. Recall [58] that a linear code $[n, k, d]$ belongs to the class of MDS codes if it has the largest possible minimum distance $d = n-k+1$.

In distributed networked storage, the combinatorial constructions in which every set of k linear combinations results in a full rank matrix are quite difficult to achieve. To better assess the involved complexity, let us imagine a centralized entity that has an access to a repository of packets produced by k sources, and has agents connected to every random subset of k storage nodes (following MDS requirement), then each such subset is in fact a coupon collector. How many times each agent needs to access the repository, or, equivalently, how many replicas should each source send to satisfy each agent, is solved using theory of perfect matching of bipartite graphs, and the related Edmunds and Erdosz theorems on matrix rank.

The result is formalized by Theorem 1 in [36]: if each source disseminate $c \ln k$ packet copies, the probability that a combination matrix is not full rank is smaller than $k/q + o(1)$ for any $c > 5n/k$, where q is the number of elements in the field. Apart from using perfect matching, the proof also uses Schwartz-Zippel theorem on the probability that a determinant of a matrix with finite field elements is equal to zero (i.e., that the randomly selected coefficients in linear combinations are the roots of the polynomial).

4.2.2 Pre-Routing: Distributed Algoritms for Random Sampling

As mentioned previously, pre-routing encompasses selecting uniformly at random a destination node from the network and providing a route (in a distributed manner) for a data packet to follow, in order to reach that destination. The key problem is that sources are not aware of the nodes in the network, except for those in their immediate neighborhood. If a source had a list of all nodes and their locations, it would draw a node from the list uniformly at random, and insert its location in the packet's header to be routed by some geographic forwarding protocol toward the selected node. Since this is not the case, energy-efficient solution for a distributed pre-routing mechanism is a significant challenge.

The algorithms and techniques for distributed pre-routing to find a random node using only local information are not discussed in [36]. One possible way to achieve it is through a random walk: start the random walk from the source node and hop randomly for a number of steps where each step corresponds to selecting a new node from the neighborhood as the new data packet relay. When the number of steps is *sufficiently large* the random walk will stop at a node *uniformly* selected from the network [64]. The uniform distribution here is a result of selecting the next hop uniformly from the set of the current node's neighbors. This is often referred to as *normal random walk*. Other forwarding rules result in other stopping distributions (not uniform). When the random walk is modeling a Markov Chain whose states are network nodes, the sufficiently large number of steps corresponds to the so-called *mixing time* of this chain, and the probability of stopping at a particular node (state) corresponds to its stationary probability.

In the next coding approach [38] that we analyze, the authors do propose a random-walk based distributed mechanism for random pre-routing in a network of location-unaware nodes, in order to achieve Fountain type storage, i.e. to ensure that the required statistics of created linear combinations of packets are achieved. The network fountain code proposed in the paper can be also represented by a bipartite graph which consists of sources as *variable nodes*, encoding sensor nodes as *check nodes*, and the edges of the graph represent the mapping between the sources and the check nodes. Note that we use the terminology related to decoding graphs used in belief-propagation decoding.

Dissemination path is a random walk trajectory a packet follows from a source until it gets stored at a check node. The authors propose a variant of Markov Chain Monte-Carlo simulation methods [65] to accomplish fast-mixing random walks over the network graph, which should ultimately converge to a desired distribution (e.g., Robust Soliton [66]) of check node degrees. The check node degree d is the number of *independent packets* that are received by the node after following properly constrained and sufficiently long random walk trajectories. The network connectivity graph considered here is the random geometric graph. Recall that random geometric graphs *(RGG)* with parameters n and r are constructed by throwing n points (nodes) randomly uniformly into the unit square and adding edges to connect any two points which are at distance at most r from each other [67].

In essence, on the network graph level, carefully designed probabilistic forwarding tables at each node result in a transition probability matrix describing Markov Chain whose stationary distribution is related to the desired code symbol degree distribution $\mu(d)$. Hence, after a sufficiently long (mixing) time, the random walks, each carrying a packet from one of k network sources, could drop the packets at random nodes, and this random process would eventually result in linear combinations of packets stored at individual nodes, and whose code symbol degree adheres to the designed distribution. The length of the random walk is proportional to the transmission cost of disseminating a source block, which must be minimized.

There is a family of Markov Chain transition probability matrices that converges to a given stationary distribution, and the fastest converging one is obtained by minimizing the second largest eigenvalue modulus, accoring to the spectral theory of matrices. For a topology modeled by random geometric graph, the length of the random walk can be approximated with the mixing time of the graph, which is $O(n \log n)$ [39].

Although in [38] the Markov Chain transition probability matrix emulated by the random walk based on a heuristic method (Metropolis-Hastings algorithm) does not induce a minimum mixing time, the algorithm is distributed and hence appropriate for large sensor networks. The transmission cost of such a dissemination, based on decentralized fountain codes, is the product of the number of random walks and the length of random walks, where the latter now only depends on the network topology. With b copies per source, and Robust Soliton code symbol degree distribution $\mu(d)$, balancing supply and demand implies that the number of random walks is

$$bk \geq n \sum_{d=1}^{k} d\mu(d), \tag{4.2}$$

since a node may need to receive more than d packets in order to store d distinct ones. Alternatively,

$$bk = n \sum_{d=1}^{k} x_d d\mu(d) > n \ln \frac{k}{\delta}, \tag{4.3}$$

where $x_d > 1$ are slack variables that can be calculated from the constraints on how much the probability of degree d can diverge from Robust Soliton.

Recall that in [36] each source disseminates independently, with the relaxed MDS requirement that the probability of decoding from any set of k codewords is high enough. Even though this scheme does not aim to satisfy a given code symbol degree distribution, it provides a tighter bound on the number of per-source replicas b needed for the scheme in [38], than the one given by (4.3). Hence, if linear decoding complexity is desired, and we decide to pursue approach given in [38], loosely bounding b by the number of copies per source required by [36] ($b > 5n \ln k/k$), then, for a network of $n = 10^4$ nodes, and where at any time 10% of nodes are sources, more than 200 random walks per source are needed. Obviously, a stricter requirement imposed on implementations of decentralized Fountain codes than just high probability of decoding from any set of k codewords (in terms of satisfying code symbol degree statistics) makes the above bound very loose.

Another recent work on distributed storage based on decentralized Fountain codes [39, 40] avoids sending multiple replicas from each source in order to satisfy code symbol degree statistics by using additional processing of received packets. Yet, in order to properly mix the data across the network, sufficiently long random walks allow packets to reach all nodes. Each network node needs to maintain k bits to keep track if it was already visited by a particular random walk. The stored linear combination of packets may or may not be modified by a first-time passing packet, as a result of a Bernoulli experiment performed with success probability $p = \frac{d}{k}$. Here, d is the node's desired code symbol degree drawn from Robust Soliton. Despite the fact that only one random walk per source is initiated, the length of random walks $O(n \log n)$ incurs high communication cost.

4.2.3 Data Recovery

On a positive note, the higher dissemination cost in decentralized Fountain schemes [39, 40, 68] results in a better control of the degree distribution, which allows for belief propagation decoding of the stored packet combinations. With belief propagation, decoding complexity for Fountain codes is $O(k \ln k)$. Any erasure code can be decoded using Gaussian elimination in $(O(k^3))$. Hence, decentralized erasure codes, such as the variant described in [36], as well as random linear coding, have decoding complexity $(O(k^3))$. Exploiting the sparsity of the linear equations can result in faster decoding: with the Wiedemann algorithm [69] one can decode decentralized erasure codes in $O(k^2 \log(k))$ time on average.

The issue of decoding complexity is associated with the partial recovery problem, where one is interested in querying fewer than k nodes and recovering partial information. So far we have been addressing the problem of recovering all k data packets by querying k storage nodes. With a rapidly failing sensor network, the queries need to be adjusted to network dynamics since some nodes are sensing information that needs to reach the data collectors as soon as possible. Sanghavi [70] investigated the optimal degree distribution for fountain codes when one is satisfied by recovering less than k data packets. Upper bounds on the performance of any degree distribution and lower bounds achieved by optimized distributions are presented for different fractions of k. To an extent, although in a different setup, the results of Sanghavi's paper are confirmed by a practical distributed storage scheme proposed in [35].

This paper [35] introduces Growth Codes with a dynamically varying degree distribution. Here, we have a classical network coding mechanism where nodes exchange code symbols with their neighbors and combine received code symbols with the existing local information before storing it. As a node initializes the memory with its own symbol and randomly picks code symbols to be transmitted from memory, the probability of transmitting own information is initially very high and then gradually decreases as the memory is filled with other code symbols. Hence, the code symbol degree gradually increase with time, as data from different sources gets mixed in the network. This varying code symbol degree distribution optimizes sensor network data persistence under node failure, as it allows, with high probability, for partial recovery of the information corresponding to any subset of nodes. Intuitively, a high degree increases the probability that code symbol transmissions are innovative, while a low degree increases the probability that the information can be decoded immediately upon reception.

4.3 Critical Discussion of the Existing Solutions

In conclusion of this review, let us point out to two common features of the presented encoding solutions for in-network distributed data storage:

Costly Random Uniform Sampling of Network Nodes: Every source node performs this sampling of nodes when selecting a node to store its packet at. Such uniformly-random storage selection is needed to make the data available for collection everywhere (i.e. from a random subset of nodes), and imposes a dissemination method that would properly ”mix” data around the network graph. Note that this is a slightly different issue than achieving a particular desired distribution of packet loads stored accross the network (in terms of the number of distinct packets encoded per node, i.e. code symbol degrees), such as in [38]. In fact, if the desired distribution of the codeword degrees were uniform (and we only send one packet replica from each source), it would represent the same requirement. Then, we would have equal chances to decode k original packets from a set of storage nodes in any part of the network, but this set would have to be much larger than k.

As we already pointed out, dissemination methods to properly mix data across the network are based on random walks, one alternative being simple flooding, which we do not consider for obvious reasons. Minimum mixing time of random walks [71] provides a measure of optimality for the lengths of the dissemination paths. The minimum mixing time depends on the network topology graph, and unfortunately, for realistic sensor network topologies (modeled by random geometric graphs), this mixing time is very large ($O\left(n \ln n\right)$). We find this observation very important, as it implies that the transmission cost might be further decreased, if the network graph is modified through topology control, or established by building an infrastructure of spokes. In fact, in the proposed approach, we rely on the symmetry of the circular subgraphs within the infrastructure of spokes to create deterministic, yet distributed, dissemination mechanisms, resulting in decreased communication cost.

Note that data mixing mechanisms are not required for Growth Codes, as these codes are based on the assumption of an extremely dynamic network topology (with mobile nodes, for example). Nodes encounter other nodes with uniform probability, i.e., the neighborhood of a node when transmitting a code symbol is uncorrelated with the neighborhood during previous transmissions. In less random scenarios, coding performance will be sub-optimal. We conjecture that an overlay network (e.g. based on BeSpoken infrastructure) which can be utilized to forward data not only to nodes in the geographic proximity, but also along long-haul paths, can provide the effect of a dynamic network topology.

Ignoring Broadcast Nature of the Wireless Medium: Let us first broadly comment on random network coding in the context of the broadcast nature of the wireless medium. Here, the links cause packets to be spread about in probabilistic manner, which is often referred to as *wireless multicast advantage*. Consequently, there is no reason to restrict information flow to a path as in wireline networks, because every node in the network can potentially act as a relay, encoding packets it receives and sending out these encoded packets. A significant observation is that none of the presented dissemination mechanisms (used to mechanize decentralized encoding) leverage the wireless multicast advantage. Random walk prerouting relies on forwarding a packet to one of the neighbors in the network graph, while the fact that all the neighbors are overhearing the same transmission is not considered. In contrast, the decentralized storage that we propose next is aiming to incorporate the wireless multicast advantage into the dissemination model.

Chapter 5

BeSpoken Network Infrastructure for Data Collecting

In this chapter, we discuss a BeSpoken network infrastructure for data collecting, and propose new strategies to collect Fountain-encoded data stored within such infrastructure. To illustrate the scalability of data collecting facilitated by the existence of such infrastructure, we also provide several heuristic methods for coding-based data storage, while a detailed description of an architecture for distributed storage and data collection, based on a BeSpoken subnetwork, is given in the next chapter, followed by a thorough analysis and a discussion of performance gains.

As discussed in Chapter 1, the BeSpoken infrastructure provides a set of data paths for dissemination, and a virtual coordinate system for data querying and localization. We propose to employ distributed coding and decoding methods to enable scalable data dissemination and storage within the proposed infrastructure. The BeSpoken-based solution is motivated by the fact that the classical dissemination methods suffer from excessive complexity that cannot be sustained by simple sensor nodes: conventional shortest-path routing algorithms are not scalable as they require each sensor to maintain a routing table with a size proportional to the total number of sensors in the network, while geographic routing protocols [12, 25], although more scalable, assume that sensors know their respective locations. Next, proposed dissemination methods suggest network-wide storage of linear combinations of data from the source nodes, hence making it possible for the data collector to visit only a sufficiently large random (and local) subset of network nodes to collect all the data. Data collecting efficiency is measured through the resources required to distribute the data from the source nodes, the number of storage nodes that the data collector needs to visit and the effort (number of operations) required for both encoding and decoding of data. In the previous chapter, we argued that the existing distributed coding methods are not efficient and, in this and the following chapter, study how a

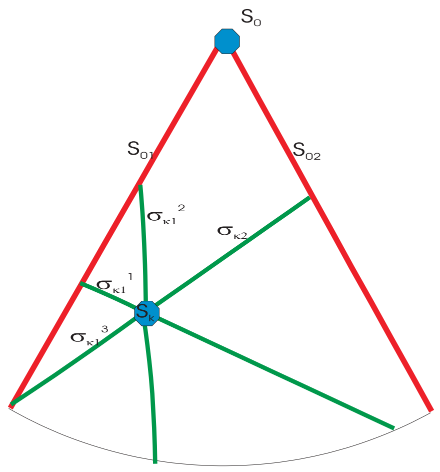
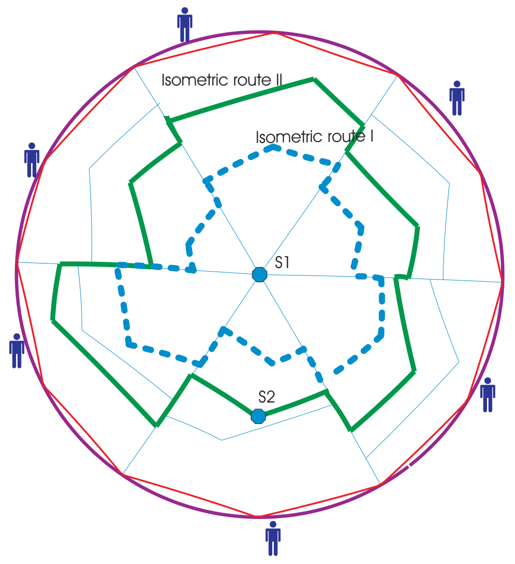


Figure 5.1: A complete set of source spokes during infrastructure-building phase. Both source localization and infrastructure mapping are illustrated here: Backbone spokes S_{01} and S_{02} are already known (directions and lengths). S_k infers that out of three spokes intersecting the backbone spoke S_{01}, σ_{k1}^{1} is likely perpendicular to S_{01}, based on the length of the spokes. With that premise, S_k can estimate the direction of σ_{k2}, based on geometry arguments. Knowing the lengths and directions of σ_{k1} and σ_{k2}, S_k estimates its location.

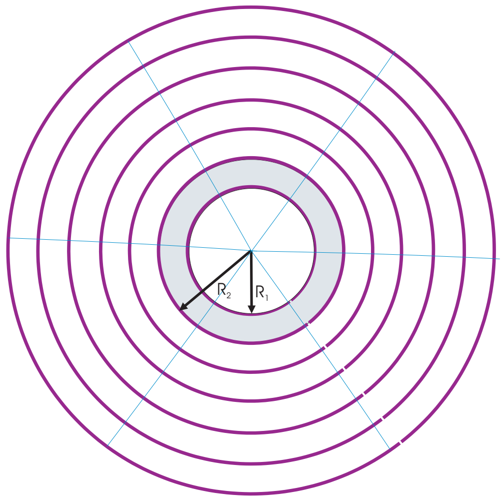
BeSpoken-based infrastructure allows for significantly improving their efficiency.

5.1 Event Localization

The events in the network are observed by the source nodes at any time after the creation of the backbone spokes. During the infrastructure building phase, the events are advertised along the source spokes, designed to be shorter than backbone spokes. Thus the transmission range of source spoke leading relays r_s is smaller than r, i.e. $r_s = gr, g < 1$. These source spokes are equally spaced around the source and incrementally grown until the two closest adjacent backbone spokes are intersected, as shown in Fig. 5.1. Each spoke is extended by a fixed number of hops at a time. If an intersection occurs due to this incremental extension, the nodes of the intersecting spoke relay the alert back to the source. Upon the first intersection event, the source stops the growth of the pertaining spoke only, while the subsequent intersection with the adjacent backbone spoke is signaled to all other spokes from the same source set, to stop further propagation. If the intersection has not been announced, the spokes get extended by the next round of hops. The resulting set of source spokes is shown in Fig. 5.1. We here focus



(a)



(b)

Figure 5.2: **(a)** Transversal spokes form associations at the common intersecting backbone spoke creating the ordered routes, according to some measure of distance from the center **(b)** The number of sources associated with a particular isometric route is proportional to the area of the annulus outlined by this isometric route and the adjacent one, closer to the centre

on the two spokes intersecting the adjacent backbone spokes, which we refer to as transversal spokes (Fig. 1.2 (b)). Other source spokes are important for source localization (Figure 5.1), and may be used to enhance the impact of various infrastructure-based WSN solutions, but we do not analyze their contribution here.

5.2 Isometric Networks

Let us assume there is a sufficiently large number k of source nodes. Two transversal spokes from adjacent wedges, that are on a similar distance from the network center, associate with each other to form an isometric circular route. The following procedure is exercised: at each backbone spoke intersection, the source whose spoke is forming the intersection looks for the adjacent wedge spoke that forms the next upstream (closer to the central node) intersection with this backbone. If such an intersection is not available, then it looks for the next downstream intersection. In either case, the initiating source makes an association with the source of the pertinent intersecting spoke, forming a route that goes through its own transversal spokes, part of the backbone spoke and the transversal spokes of the associated source. The process continues from wedge to wedge, and closes into a circular route that we call an *isometric route* (Fig. 5.2 (a)).

We assume that there are six wedges, and the same number of sources instantiating each isometric route. However, there may be more sources in a wedge per isometric route, as sources that are one (or a limited number) hop away from the existing source spokes do not create their own spokes but utilize the closest isometric route to distribute their packets. Thus, each isometric route I_i creates a "one-dimensional network" N_i in itself. Assuming that isometric routes are perfect circles of radius R_i, and considering uniform distribution of events $\lambda_s < 1$ (for unit density of nodes) , the number of sources associated with a particular isometric route is proportional to the area of the annulus outlined by this isometric route and the adjacent one, closer to the centre, as shown in Fig. 5.2 (b). Thus, the expected number of sources in N_i is

$$\bar{K}_i = \left(R_i^2 - R_{i-1}^2\right) \pi \lambda_s, \tag{5.1}$$

where $i \geq 1$ and $R_0 = 0$, while the expected number of nodes in this network is approximately equal to the area of the annulus $2\pi R_i r_s$, assuming the unit density of nodes. We allow for the possibility of an open isometric route, since it does not change the logic of our model. The isometric routes are ordered by their proximity to the central node. The last isometric route is created differently, and it is identical with the perimeter route.

5.3 Push-Pull Model of Data Collecting Based on BeSpoken Infrastructure

In the previous chapter we studied distributed solutions for the storage protocol that makes the entire data set of k source packets available in any collection of $k\,(1+\epsilon)$ storage nodes, accessible by a data collector who approaches the network perimeter at an unknown location (see Figure 1.2). Here, we introduce a data collection architecture which assumes that such storage schemes are available in a network whose nodes implement BeSpoken protocol.

There is a fundamental tradeoff between network storage capacity and the collection delay. If each node across the network can store all the packets, the data can be collected in a single hop. On the other extreme, if each node has own data destined for the collector and no capacity to store other packets, the collector has to reach out to all the network nodes to collect the data, which would incur an extreme delay. The canonical model considered here is when the number of source nodes k is smaller than the number of network nodes and where each network node can both relay and store one (possibly encoded) packet.

More efficient storage codes than the simplest repetition code require that the collector not only collects the linear combinations but also is capable of decoding/recovering the original packets. A well known codes that are suited for belief propagation decoding, characterized by linear decoding complexity, are LT codes [66].

Given BeSpoken infrastructure, we now propose a data collecting approach which relaxes the decoding requirement based on the asymptotic analysis of LT codes [66], which demands that $k + \sqrt{k}\log^2(k/\delta)$ code symbols be collected to decode $(1-\delta)k$ original symbols. Here, δ is a sufficiently small constant. In the proposed strategy, the collector decides to collect *only* k linear combinations before it starts the decoding.

We assume that random linear combinations are created along the infrastructure that can be forwarded down the backbone spokes to the perimeter nodes in the proximity of an intelligent collector, as soon as the collector announces itself by sending a query along the same backbone spokes. Hence, the encoded data, with the code symbol degree statistics that are congruous with belief-propagation decoding, can be gathered from the perimeter nodes in the collector's vicinity. Then, as the belief propagation decoding stops, possibly with a couple of packets left undecoded (as we collected less than demanded by the bound in [66]), the collector can query the nearby infrastructure elements for the original pieces or independent combinations that are missing. This is possible since each coded packet contains a header with d source packet IDs, where d is the code symbol degree.

In this manner, a push approach, resulting in in-network storage, and facilitated by network coding, is combined with a pull approach (searching for particular data pieces). The push element allows for the data availability (low access delay) and persistence, while, hopefully, also decreasing communication cost of data collection. The lower energy expenditure due to decreased communication is made possible by the subsequent pull element of data dissemination, which heavily relies on the BeSpoken infrastructure properties, and the nature of belief-propagation decoding. We devote majority of the remaining chapter to study energy/collection-delay efficiency of the presented model.

5.4 Heuristic Storage Methods Along Isometric Networks

We here identify simple subgraphs within the infrastructure, namely one-dimensional, circular graphs of isometric networks N_i, each with K_i sources, and devise useful heuristics for distributed coding over associated subnetworks. We are motivated by the outlined deficiencies of the existing decentralized methods for encoded storage, and by the observation that the BeSpoken infrastructure represents a graph overlaid on top of the sensor network graph, and amenable to further topology control. This potentially allows for producing a graph with a minimum mixing time that is much shorter than the mixing time of a random geometric graph (unit graph) [72], typically used to model the WSN topology.

Before presenting some examples of how the infrastructure can be utilized for distributed coding over simple subgraphs, let us first introduce some terminology. We have already associated one-dimensional isometric networks N_i with isometric routes I_i, and now we define two classes of isometric networks, based on the number of sources $\bar{K}_i$ expected to be supported by such a network. If $\bar{K}_i$ (5.1) is larger than a given threshold, the isometric network is considered *heavy*, and in the opposite case we refer to it as *light*. For large number of sources, it is computationally more efficient for data collector to decode fountain encoded stored data. Hence, heavy networks will be properly parametrized to store data by following a distributed protocol that is expected to result in such a decentralized fountain code. Conversely, light isometric networks will employ a variant of decentralized erasure codes to store data generated by smaller number of sources.

Now, to be able to describe the notion of hubs, let us recall that backbone spokes are not only sequences of leading relays; all nodes in the transmission range of a leading relay are aware of belonging to the spoke. In other words, a spoke is a strip of certain width, not a line. As an isometric route intersects a backbone spoke, it creates an implicit route to the set of nodes in data transmission range (disk of radius r) of the leading relay belonging to this intersection. This set of nodes is referred to as *storage hub* (see Figure 5.3(a)). Storage hub H_{ij} is indexed by the backbone spoke index i and the isometric route index j. Sources in the isometric network N_j advertise and store packets with the pertaining storage hubs. More accurately, light isometric networks use hubs for storing data, which we describe soon, while both heavy and light ones use it as an advertising hub, in the following manner.

Let us first point out that in all distributed networked storage examples presented in Chapter 4, except for a variant in [39], the number of sources k is known ahead of time. In [39], a heuristic method is used to estimate k and n, although its communication cost may be prohibitive, as it requires additional random walks of length $n \log n$ during the pre-coding phase. With BeSpoken architecture, these numbers do not have to be fixed. Once all the sources are connected to the backbone through isometric routes, at any moment we have a population of K active sources, where K is a random number, and $K = \sum_i K_i$. A hub can periodically request from active sources to announce themselves, or the sources could be propagating unsolicited data advertisements around their isometric route. Either way, the storage hubs act as counting hubs. Each hub should update the count of sources as a new ad is encountered. At some point, the hub broadcasts a confirmation to all the sources associated with the isometric route and they start sending data.

We currently assume there is one dedicated hub per isometric network at a time. The hub is aware, by the estimated distance of the isometric route from the centre, if its isometric network is likely to be light, and prepares itself to store encoded network data, if that's the case. The hub can always switch the network to a different class, if the reported number of data is higher than the threshold. Threshold is established not only based on the number of sources, but also on the number of nodes h in the hub. For fountain codes, this number would need to be large enough compared to the current number of sources $K_i = k_i$, to guarantee that sampling from the Robust Soliton will result in an empirical distribution of codeword degrees that is close to Robust Soliton. The leading relay inside the hub transmits the count K_i message to the hub nodes. The hub members know how many of them are in the hub (based on previously send "Hello" messages to the leading relay). Based on (5.1), while assuming that, on average, there will be one-to-one mapping between each leading relay in a backbone spoke and an isometric route, which (on average) is equivalent to ith isometric route having radius $R_i = ir$, we obtain the following law

$$\begin{aligned}\bar{K_i} &= r^2\left(i^2 - (i-1)^2\right)\pi\lambda_s \\ &= r^2\pi\lambda_s\left(2i-1\right),\end{aligned} \tag{5.2}$$

describing the expected number of sources in isometric network N_i as a function of its distance from the central node. As the expected number of nodes in each hub $\bar{h} = r^2\pi$, we can find the percentage of network nodes that are likely to use a hub for storing its encoded data.

5.4.1 BeSpoken Light Isometric Networks: Random Linear Coding

Before the leading relay of a hub sends a query for data along its isometric route, each hub node sets the probability parameter $p = c\ln K_i/h$, where $c = 5h/K_i$ by Theorem 1 in [36], given that the isometric network is light. The parameter p is smaller than one, with high probability, as $5\ln K_i < K_i$ for $K_i > 12$, while, for the network sizes that we consider, the expected number of nodes in the ith hub $\bar{h}$ (based on 2.5) is significantly larger than 12 and calls for repetition coding of such small number of source packets. The repetition code here means that each hub node will choose to store one of the K_i packets with probability $1/K_i$, so that each source packet gets stored in h/K_i nodes in average. Note that the variance of h is not large, by the virtue of the spoke building process that "avoids" network holes.

Let us now go back to typical light isometric networks with $K_i > 12$. The probability parameter is chosen so that, if each of h nodes stored a packet with this probability, after k_i packets the hub will have stored $ck_i\ln k_i$ packets in average, which is enough to claim that there will be k_i independent linear combinations inside the hub, based on the result in [36].

The packets are forwarded along the isometric route. They reach the hub's leading relay after $n_i/4$ hops in average, and the relay retransmits it to the hub members, with the transmission range r (note that previous hops along the isometric route were of different range).

Now, each node flips the coin with probability p whether to store the packet, or to drop it. Here, we assume that each node has unit memory, and if a node's memory is m units, we count it as m nodes. If the node chooses to store a packet while it already has a packet

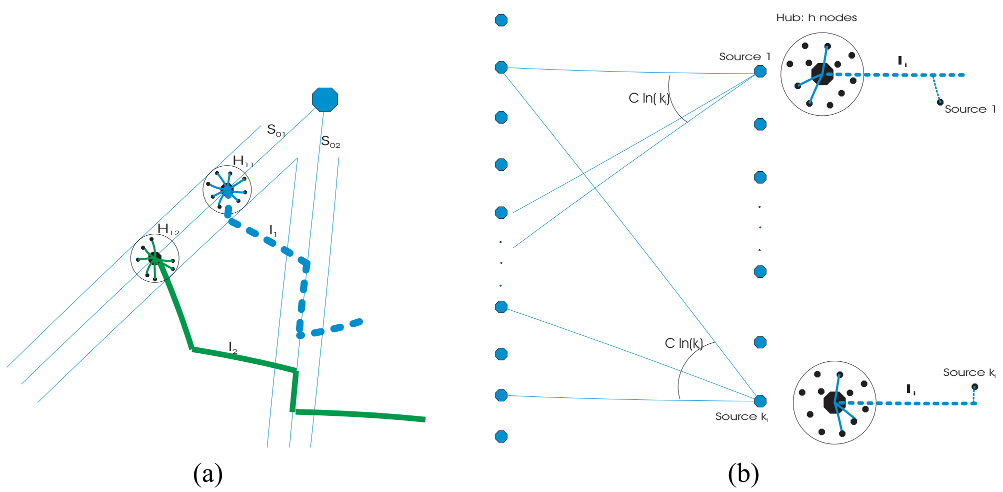


Figure 5.3: **(a)** For each source belonging to the isometric network N_i, the expected number of packet replicas stored in a hub H_{xi}, where x denotes a backbone spoke, is $\ln k_i$ **(b)** An example of the coding graph for the light isometric network N_1, with the associated isometric route I_1, whose hub H_{11} is shown in (a)

stored, it creates a linear combination of the two over a $GF(q)$, with coefficients randomly selected from the same field. Based on Coupon Collector model, after all K_i packet types are stored in the hub, and the expected $cK_i \ln K_i$ packets have been combined, we expect to have K_i independent linear combinations of the packets which is enough to decode with high probability. This dissemination and coding method is illustrated in Figure 5.3. As soon as the data collector appears and sends a data request along the backbone spoke, the hub nodes start sending code symbols down the spoke for the collector to decode.

Since this is a variety of random linear codes, the decoding complexity for the light isometric network N_i is $O(K_i^3)$. If there are K sources in the network, and all the isometric subnetworks were light, the decoding complexity of $O(\sum_i K_i^3)$ is already better than $O(K^3)$, which would be the complexity for the same coding approach applied to the whole sensor network.

5.4.2 BeSpoken Heavy Isometric Networks: Distributed Fountain Coding

In case of heavy isometric networks, the idea is to implement a variant of distributed fountain codes, since the number of sources K_i supported by such a network (indexed by i) is expected to be large, and random linear coding as applied to light isometric networks would result in high-complexity decoding ($O(K_i^3)$). The approach for delivering encoded data is the same as with light networks: upon the data collector's query, certain isometric network storage nodes deliver encoded data by sending them down the backbone spoke. The optimal distribution of storage nodes along the isometric route is one of the key questions so that the $O(K_i)$ storage nodes can be in the "sensitivity range" around the affected backbone spoke and respond to the query efficiently. Another key question is the coding method.

To each heavy isometric network N_i we associate an one-dimensional (circular) dissemination graph whose n_i nodes are leading relays of the composing source spokes. Thus, as the perimeter of the ith isometric route is $2\pi ri$, and the (maximum) hop length between the leading relays of the source spokes is $r_s = gr, g < 1$, the number of nodes in the graph is about $n_i = (2\pi i)/g > 2\pi i$. As we envision to have isometric networks that are farther away from the central node configured as heavy, let us assume that $i > 10$. By 2.5, the spoke hop length designed for networks of relevant sizes indicates that for networks larger than 10000 nodes we will have more than ten isometric routes (i.e. more than ten hops allong a backbone spoke), with high probability. Each node in the one-dimensional graph is connected to one node to the right and one node to the left.

Let us assume that packets are passed from one leading relay to another, following a simple random walk, where the probability of choosing the next hop on either side of the current node is one half. According to [72], the mixing time of a random walk over one-dimensional graph of n_i nodes, where each node is connected to f nodes to the right and f nodes to the left, and $f \leq n_i/4$, should be of order $O(\ln n_i/(1-\cos(2\pi f/n_i)))$. For $i = 10$ and $f = 1$, this gives the mixing time T_i of order

$$\begin{aligned} T_i &= O(\frac{\ln(2\pi i)}{1-\cos(2\pi/(2\pi i))}) \\ &= O(\frac{\ln(2\pi i)}{1/i^2}) \\ &= O(i^2 \ln i), \end{aligned} \tag{5.3}$$

where we replaced $\cos(x)$ with the Taylor expansion $\cos(x) = 1 - x^2/2 + O(x^4)$.

Now, such a random walk is too costly to be used as a pre-routing mechanism employed in distributed Fountain encoding akin to [38–40]. Moreover, the circular graph used here as a model of the isometric network does not reflect the multicast nature of the wireless medium.

In the next chapter, (Figure 6.3) we introduce another model of the isometric network that does incorporate wireless multicast advantage, and also design a pre-routing protocol that utilizes the symmetry of this graph instead of letting source packets follow random walk trajectories.

More precisely, a detailed model of data gathering in heavy isometric networks, which includes all aspects of data dissemination and distributed storage, is the subject of the following chapter.

5.5 Conclusions

In the previous two chapters we discussed the existing decentralized coding-based storage methods, and analyzed the potential of the BeSpoken-based infrastructure to mitigate some of the drawbacks these methods have. We started from the fact that we have the BeSpoken infrastructure ”for free”, assuming that it has been built for the purpose of searching for particular data instances, by using the model of intersecting source and sink spokes, and, in addition, for localization of network events, through a virtual coordinate system, based on the hop-distance from the backbone spokes. As we highlighted in the subsection 4.3, the observed deficiencies of the current approaches to network coding for data persistence and in-network data storage in WSNs are a great deal due to the topology of the dissemination graph, typically modeled as random geometric graph. The BeSpoken infrastructure offers a flexibility in terms of partitioning the network into regular subgraphs over which we can achieve better coding efficiency. This efficiency is measured by several parameters:

- **communication cost of data dissemination paths, which affects the total energy consumed by the network to store easily collectable data.** Here, we plan to express the energy savings as a result of both partitioning (as $\sum_i k_i \ln k_i < k \ln k$, for $k = \sum_i k_i$), and a tradeoff of decoding requirements, achieved through the push-pull model of data collecting.

- **decoding complexity, which affects the delay in reconstructing network data at the data collector.** Here, we use partitioning to achieve shorter decoding time for light netwoks (as $\sum_i k_i^3 < k^3$ for $k = \sum_i k_i$), and we use push-pull method for heavy networks, to combine partial decoding with specific data queries. We next present a detailed complexity analysis of the proposed decoding model.

- **amenability to a distributed and ad-hoc implementation (e.g. the number of sources does not need to be known ahead of time).**

- **scalability.** A good practice in treating scalabilty issues is to introduce network hierarchy, or network partitioning. Here, network is partitioned into subnetworks that are customized to handle network coding task according to number of the associated sources. In the next section, we attempt to establish specific measures of scalability for both the communication and the decoding-complexity aspect of the proposed data collecting strategy.

We address many of these aspects of efficiency in the discussion of our Fountain-based storage and collection strategy, presented in the next chapter.

Chapter 6

Doped Fountain Coding for Minimum Delay Data Collection

6.1 Introduction

In this chapter we analyze decentralized Fountain-type network coding strategies for facilitating a reduced delay data collection and network coding schemes for efficient data dissemination for a planar donut-shaped sensor network, introduced in previous chapter (see also Figure 6.1), whose nodes lie between two concentric circles. The network backbone is a circular route of relay nodes which disseminate data. All network nodes within its transmission range overhear relay's transmissions and serve as potential storage nodes. The storage nodes within a relay's transmission range form a squad. The squad size determines the relay's one-hop storage capacity. Squad's storage capacity together with the source node density and the coding/collection strategy determine the data collection delay measured in terms of the number of communication hops required for the collector to collect and recover all k source data packets. In the proposed polling (packet doping) scheme, an intelligent data collector *(IDC)* first collects a minimum set of coded packets from a subset of storage squads in its proximity, which might be sufficient for recovering the original packets and, by using a message-passing decoder, attempts recovering all original source packets from this set. Whenever the decoder stalls, the source packet which restarts decoding is polled/doped from its original source node (at an increased delay since this packet is likely not to be close to the collector). The random-walk-based analysis of the decoding/doping process represents the key contribution of this chapter. It furnishes the collection delay analysis with a prediction on the number of required doped packets. The number of required packet dopings is surprisingly small and, hence, to reduce the number of collection hops required to collect and recover the source data one should employ the *doping* collection scheme. The delay gain due to doping is more significant when the relay squad storage capacity is smaller. Furthermore, employing network coding makes dissemination more efficient at the

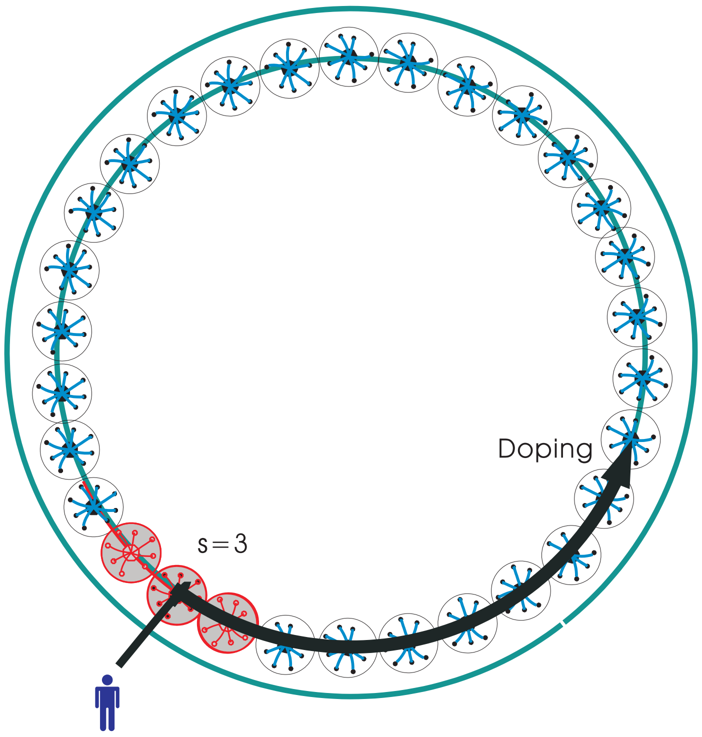


Figure 6.1: Collection of coded symbols: pull phase brings the three squads of coded packets to the decoder, and then, whenever the decoder gets stalled, an original symbol is pulled off the network.

expense of a larger collection delay. Not surprisingly, a circular network allows for a significantly more (analytically) tractable strategies relative to a network whose model is a random geometric graph [38–40].

6.2 System Model and Problem Formulation

We consider an inaccessible static wireless sensor network (e.g., a disaster recovery network) with network nodes that are capable of sensing, relaying, and storing data. As described previously, nodes are randomly scattered in a plane according to a Poisson point process of some intensity μ. The nodes have constrained memory resources. Without loss of generality, we assume that most nodes have a unit-size buffer. Each node that senses an event creates a unit-size description data packet. We refer to such a node as a data source. We assume that events are

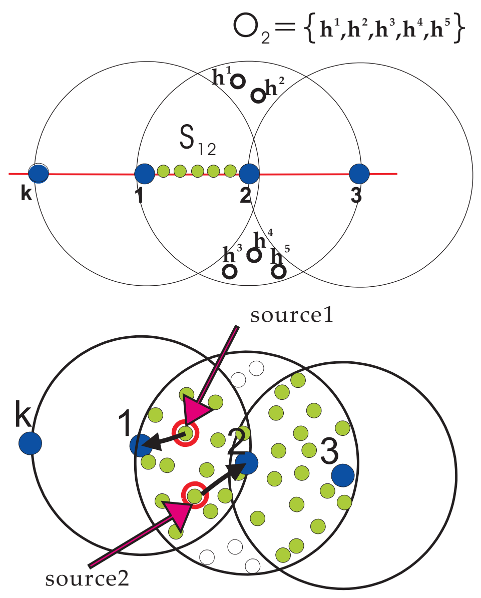


Figure 6.2: Close-Up of a Circular Squad Network of k relays. Each relay is overheard by nodes in its transmission range, referred to as squad nodes.

distributed as a Poisson point process of intensity $\mu_s < \mu$. We define the transmission range as the maximum distance r from the transmitter at which nodes can reliably receive a packet. Assuming radially symmetric attenuation (isotropic propagation), the transmitted packet is reliably received in a disk of area πr^2, illustrated in Figure 6.2. The expected number of network nodes in the disk is $\pi \mu r^2$ and the expected number of source nodes is $\pi \mu_s r^2$. Within the sensor network, we consider a circular route, composed of k nodes referred to as relays. The distance between adjacent relays is equal to the transmission range r. Without a loss of generality and for simplicity, we assume that r is selected so that only a single data source node is (expected to be) within the transmission range of a relay. That is, $\pi \mu_s r^2 = 1$. Source node observes an event and sends its data packet to the relay, making it a virtual source (see Figure 6.2). Thus, k relays form a linear network (route) with data packet i assigned to relay i, $i \in [1, \ldots, k]$. Each sensor node within the range of a relay is associated with the route via a one-hop connection to a relay. We refer to the set of nodes within the range of a relay as a *squad*, and to a node as *squad-node*. Squad nodes can hear transmissions either from only relay i or from, also relay

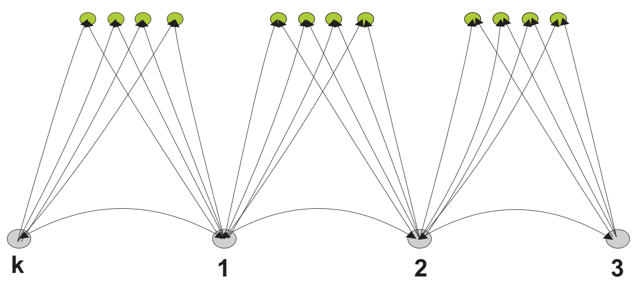


Figure 6.3: Circular Squad Network: the storage graph.

$i + 1$ and, thus, belong to either the *own set of squad-nodes* O_i, or to the *shared set of squad-nodes,* denoted $S_{i(i+1)}$, where, hereafter, any addition operation will be assumed to be mod k, i.e., $(i + 1)$ mod k, as shown in Figure 6.2. By means of associations, the relays in the circular route together with the squad nodes form a donut-shaped circular squad network. The expected number of nodes in the squad, is denoted with $h = \mu r^2 \pi$, while the expected area of each shared set is $E\left[S_{i(i+1)}\right] = h_s = 0.4h$. We primarily focus on shared squad nodes. In the rest of the paper, whenever we refer to squad-nodes, we mean shared nodes, and for simplicity we assume $h = h_s$. The goal is to disseminate data from all sources and store them at squad nodes so that a collector can recover all k original packets with minimum delay. An IDC collects data via a *collection relay*. The data is collected from k_T storage nodes of which most $k_s < k_T$ reside in a set of s *adjacent* squads, including the collection relay squad. These s squads form a *supersquad* (See Figure 6.1). The number of packets not collected from the supersquad is denoted k_d.

Note that the density of sources μ_s is dependent on the spatial characteristics of the monitored physical process, i.e., the spatial density of events. A well designed sensor network will ensure that the spatial density of nodes μ is designed to properly cover this process. When r is selected to ensure $r^2 \pi \mu_s = 1$ then $h = \mu/\mu_s$ is the coverage redundancy factor. Furthermore, for a given received signal-to-noise ratio, the one-hop transmission energy E_1 and the single hop delay τ_1 are inversely proportional to μ_s. Given this relationship, we will use the number of relay hops to quantify the collection cost per source symbol, both in terms of consumed transmission energy and in terms of incurred delay.

For a given circular route radius R, the (expected) number of relays is $k = R/r$. Hence, for a given transmission range r (or μ_s), the only degree of freedom is the coverage redundancy factor h (squad size), i.e, the network density μ. By reducing μ, we decrease the average number of nodes in a squad h. This has implications to the collection (delay and energy) cost. The supersquad consists of $s = \lceil k_s/h \rceil$ squads, and the average number of hops a packet makes until it is collected by the IDC is $(s-1)/4+1$. Hence, the smaller the μ, the larger the average collection delay $\tau_s = k_s((s-1)/4+1)\tau_1$ and the energy $E_s = k_s\left((s-1)/4+1\right)E_1$ from the supersquad. Henceforth, we will, without loss of generality, normalize $\tau_1 = 1$ and $E_1 = 1$. The key collection performance measure will be the average number of collection hops per source packet c, where $c = k_s\left[1+(s-1)/4\right]/k$ when all collected packets are from the supersquad, i.e., $k_T = k_s$.

We will comparatively consider two classes of storage/encoding strategies: in the first, the IDC collects the original packets, while in the second one the collected packets are linear combinations of the original packets and, hence, the IDC needs to decode them to recover source packets. When combining is employed, constrained by the collection delay, we consider only storage strategies which allow for decoding methods of linear complexity, i.e., the use of belief propagation (BP) iterative decoders. Based on the asymptotic analysis of LT codes [66], in case when original packets are encoded into linear combinations whose degrees follow the Robust Soliton distribution, as in [38], the expected number of collected code symbols required to decode $(1-\epsilon)k$ original symbols, where ϵ is a sufficiently small constant, is

$$k_T = k_s = k + \sqrt{k}\log^2(k/\epsilon). \tag{6.1}$$

Here the number of collected packets is significantly larger than k for small to medium k-s. Hence, collection of this many packets can be expensive, in particular when the event coverage redundancy factor h is small. Collecting a smaller number of packets *upfront* would result in a stalled decoding process. Here, we take advantage of the availability of additional replicas of source packets along the circular network, to pull one such packet off the network in order to continue the stalled decoding process. See Figure 6.1. The pull phase is meant to assist the decoding process using a technique that we refer to as *doping*. In the following, encoding

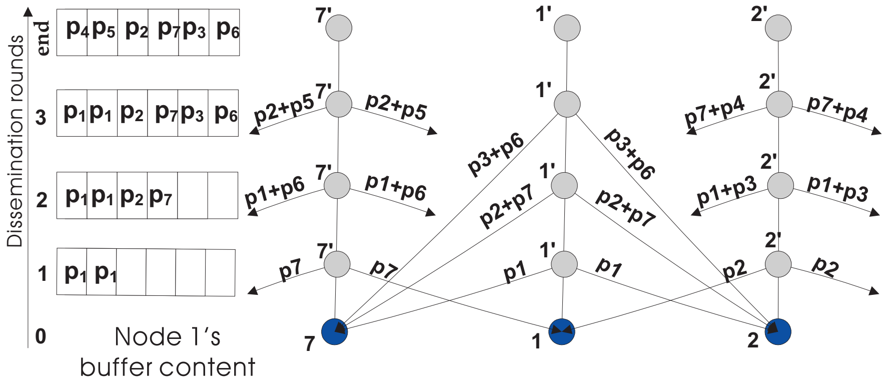


Figure 6.4: Dissemination procedure brings all network data to each relay in half as many hops as it would be needed with simple forwarding scheme: example for $k = 7$ follows the exchanges of node 1 where the black circle on the bottom represents the node's receiver while each gray circle above it represents the transmitter at the corresponding dissemination round

describes the mapping on the source packets employed both while disseminating and while storing. It is a mapping from the original k packets to the collected $k_T = k_s + k_d$ encoded packets.

6.3 Data Dissemination

The nodes within the transmission range of the route relays together with the relays themselves form a dissemination network. The dissemination connectivity graph is a simple circular graph with k nodes. This graph models connections between relays, which are bidirectional. The connectivity graph used in the storage model is expanded with storage nodes, representing shared squad nodes. In this graph, every storage node is adjacent to two neighboring relay nodes. Also, edges between storage and relay nodes are directed, as illustrated in Figure 6.3. Every edge in the dissemination graph is of unit capacity. A single transmission reaches two neighboring relays. We consider two dissemination methods: *no combining* in which each relay sends its own packet and forwards each received packet until it has seen all k network packets, and *degree-two combining*, which we describe in detail next. For the degree-two combining dissemination, a relay node combines the packet received from its left with the packet

Initialization:

k=1: Relay i sends its own packet p_i, and subsequently receives the packets $p_{(i-1)}$ and $p_{(i+1)}$ originating from its first-hop neighbors.

k=2: Relay i sends a linear combination (XOR) of the received packets $p_{(i-1)}$ and $p_{(i+1)}$, and subsequently receives the packets containing p_i XOR-ed with the packets $p_{(i-2)}$ and $p_{(i+2)}$ originating from its second-hop neighbors, respectively. Relay i recovers $p_{(i-2)}$ and $p_{(i+2)}$ by XOR-in the received linear combinations with p_i.

For $(k = 3, k < (n + 1)/2, k + +)$

Online Decoding

The packets received by relay i in the $(k - 1)$th round contain linear combination of packets $p_{(i-k+2)}$ and $p_{(i+k-2)}$ and packets $p_{(i-k+1)}$ and $p_{(i+k-1)}$, originating from its $(k - 1)$th hop neighbors. XOR-ing the received packets with the matching packets $p_{(i-k+2)}$ and $p_{(i+k-2)}$, the relays recover the packets $p_{(i-k+1)}$ and $p_{(i+k-1)}$.

Storing

The buffer space is updated with the recovered original packets $p_{(i-k+1)}$ and $p_{(i+k-1)}$. For $k > 3$ the buffer space is updated by overwriting packets $p_{(i-k+4)}$ and $p_{(i+k-4)}$.

Encoding

In the kth round, relay i linearly combines packets $p_{(i-k+1)}$ and $p_{(i+k-1)}$, and transmits the linear combination.

Figure 6.5: Degree-two Dissemination Algorithm

received from its right into a single packet by XOR-ing respective bits, to provide innovative information to both neighboring relays for the cost of one transmission [73]. Consequently, each relay performs a total of $\lceil (k-1)/2 \rceil$ first-hop exchanges, as described in Figure 6.4 and in [74]. The medium access protocol which ensures decentralized interference-free transmissions is CSMA/CA based. CSMA/CA with RTS/CTS capability ensures that if one relay obtains access to the medium, two neighboring relays on its left side, and two on its right side will not be able to transmit. Every third relay can transmit simultaneously. Thus, it takes three transmission rounds for each relay to receive their first neighbors' packets when collisions are neglected. We define these three transmission rounds as one dissemination round.

The data dissemination process starts with each relay sending its own source packet. The packets obtained from the first neighbors are stored in the relay's buffer. In the subsequent dissemination rounds, each relay creates a linear combination over GF(2) of the received packets by simply XOR-ing matching bits, and transmits the generated packet. Figure 6.4 illustrates storage requirements together with dissemination graph showing transmissions between relay 1 and its first-hop neighbors, according to the algorithm 6.5, with $k = 7$ relay nodes. The darker nodes of the dissemination graph $i \in [1, 2, 7]$, correspond to relay reception and the lighter nodes $i' \in [1, 2, 7]$ correspond to relay's transmission for each of the $(k-1)/2 = 3$ dissemination rounds. At dissemination round k, a relay decodes its k-th neighbor packets and transmits a linear combination of its $(k-1)$st neighbor packets. Six units of relay memory space are sufficient to enable this algorithm.

Note that here storage nodes overhear degree-two packet transmissions. They either randomly combine those with previously received degree-two packets, or they first apply the online decoding of the packets (see Figure 6.4), and then combine obtained degree-one packets with previously stored linear combinations of degree-one packets.

6.4 Decentralized Squad-Based Storage Encoding

Under a centralized storage mechanism that would allow coordination between squad nodes, a unique packet could be assigned to each of k nodes located within a supersquad of an approximate size k/h, and the same procedure repeated around the circular network for each set of k adjacent squad nodes. This periodic encoding procedure would allow a randomly positioned IDC to collect k original packets from the set of closest nodes. However, our focus are scalable designs where centralized solutions are not possible. We resort to stochastic protocols for storing packet replicas, and apply random coding to store linear combinations of the packets. For each dissemination method we distinguish: *combining* and *non-combining* decentralized storage techniques. In both we assume that the storage squad nodes can hear (receive) any of the k dissemination transmissions from the neighboring relay nodes. Hence, either a common timing clock or/and regular transmission listening *is* necessary.

The reference example of non-combining methods is *coupon collection* storage, in which each squad node randomly selects one of k packets to store ahead of time. As the coupon collector is completely random, it requires on average $k \log k$ storage nodes to cover all the original packets. In order to decrease the probability of many packets not being covered, we apply combining storage techniques in which one storage node's encoded packet contains information that covers many original packets. The higher this *code symbol degree* is, the lower is the likelihood that a packet will stay uncovered. We consider combining either degree-two or degree-one packets. Each squad node samples a desired code symbol degree d from distribution $\omega(d),\, d \in [1, \cdots, k]$. The squad node decides ahead of time which subset of d transmissions it will combine to generate the stored encoded packet. Choosing a good distribution $\omega(d)$ is not easy, since it needs to satisfy many contradicting requirements. The high-degree code symbols are good for decreasing the probability of uncovered packets. However, other requirements are more important for proper behavior of the BP decoding process, especially the right amount of degree one and degree two code symbols. It is well known that the expected behavior of the Ideal Soliton *(IS)* distribution, defined as

$$\rho(d) \quad = \quad \begin{cases} \frac{1}{k}, & d = 1, \\ \frac{1}{d(d-1)} & d = 2, \cdots, k, , \\ 0 & o.w. \end{cases} \tag{6.2}$$

is close to ideal for Fountain codes decoded by a BP decoder [66]. However, the BP process can be frequently stalled due to the absence of degree-one symbols (*the ripple*) in the collected sample of code symbols since the size of the ripple is a random variable of large variance and the expected value equal to one. This is the reason why Robust Soliton *(RS)*, defined as

$$\mu(d) = \frac{\rho(d) + \tau(d)}{\sum_i \rho(i) + \tau(i)}, \tag{6.3}$$

where $R = c \ln(k/\delta)\sqrt{k}$, for some positive c, and small positive $\delta < 1$, and

$$\tau(d) = \begin{cases} \frac{R}{dk}, & d = 1, \cdots, k/R - 1 \\ \frac{R\ln(R/\delta)}{k}, & d = k/R, \\ 0 & d = k/R + 1, \cdots, k \end{cases}, \tag{6.4}$$

is used as a choice degree distribution for rateless erasure codes. For RS, the probability of degree-one symbols is overdesigned in order to prevent stalling. However, redistribution of

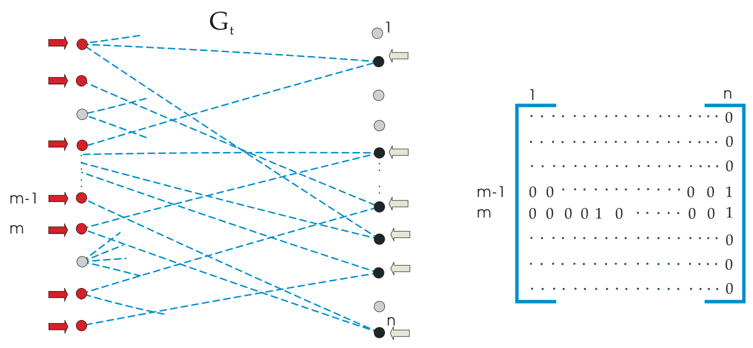


Figure 6.6: In the graph $\mathbf{G_t}$, representing the stalled decoding process at time t, we identify nodes on the left side (input symbols corresponding to rows of the incidence matrix) connected to right-hand-side nodes of degree two (output nodes corresponding to columns of weight two, represented by black nodes, and pointed to by black arrows), and then uniformly at random select one such input symbol to unlock the decoder. The set of symbols we are selecting from is represented by red nodes, indicated by red arrows.

the probability mass from higher degrees to degree-one increases the likelihood of uncovered packets. In the next section, we present an analysis of why IS turns out to be better than RS when BP doping is used.

6.5 Collection and Decoding

The collection problem with the coupon collector (and with similar non-combining storage methods) is straightforward as it excludes decoding. The focus is simply on providing coverage redundancy h that minimizes the size of supersquad containing $k \log k$ packets required to recover k source packets. For the Fountain-based combining methods, the collection problem is more elaborate, and intricately tied to decoding strategy, which we study in the following subsections.

6.5.1 Belief Propagation Decoding

Suppose that we have a set of k_s code symbols that are linear combinations of k unique input symbols, indexed by the set $\{1, \cdots, k\}$. Let the degrees of linear combinations be random numbers that follow distribution $\omega(d)$ with support $d \in \{1, \cdots, k\}$. Here, we equivalently use $\omega(d)$ and its generating polynomial $\Omega(x) = \sum_{d=1}^{k} \Omega_d x^d$, where $\Omega_d = \omega(d)$. Let us denote the graph describing the (BP) decoding process at time t by $\mathbf{G_t}$ (see Figure 6.6). We start with a decoding matrix $\mathbf{S_0} = [s_{ij}]_{k \times k_s}$, where code symbols are described using columns, so that $s_{ij} = 1$ iff the jth code symbol contains the ith input symbol. The number of ones in the column corresponds to the degree of the associated code symbol. Input symbols covered by the code symbols with degree one constitute the ripple. In the first step of the decoding process, one input symbol in the ripple is processed by being removed from all neighboring code symbols in the associated graph $\mathbf{G_0}$. If the index of the input symbol is m, this effectively removes the mth row of the matrix, thus creating the new decoding matrix $\mathbf{S_1} = [s_{ij}]_{(k-1) \times k_s}$. We refer to the code symbols modified by the removal of the processed input symbol as output symbols. Output symbols of degree one may cover additional input symbols and thus modify the ripple. Hence, the distribution of output symbol degrees changes to $\Omega_1(x)$. At each subsequent step of the decoding process one input symbol in the ripple is processed by being removed from all neighboring output symbols and all such output symbols that subsequently have exactly one remaining neighbor are released to cover that neighbor. Consequently, the support of the output symbol degrees after ℓ input symbols have been processed is $d \in \{1, \cdots, k - \ell\}$, and the resulting output degree distribution is denoted by $\Omega_\ell(x)$. Our analysis of the presented BP decoding process is based on the assumption that the ripple size relative to the number of higher degree symbols is small enough throughout the process. Consequently, we can ignore the presence of defected ripple symbols (redundant degree-one symbols) [75]. Hence, the number of decoded symbols is increased by one with each processed ripple symbol. Now, let us assume that input symbols to be processed are not taken from the ripple, but instead provided to the decoder as side information. We refer to this mechanism of processing input symbols obtained as side information as *doping*. Doping for improved decoding was first described in [76] as a technique that enables iterative decoding of serially concatenated codes. [77] reports a two-stage scheme where a code is sent in the first stage, while in the second stage the encoder maintains a dialog with belief-propagation decoder, enhanced by a doping algorithm. Here, to unlock the belief propagation process stalled at time (iteration) t, the degree-two doping strategy selects the doping symbol from the set of input symbols connected to the degree-two output symbols in graph $\mathbf{G_t}$, as illustrated in Figure 6.6. Hence, the ripple evolution is affected in a different manner, i.e. with doping-enhanced decoding process the ripple size does not necessarily decrease by one with each processed input symbol.

The following subsections study the behavior of both varieties of the BP decoding process, first through the evolution of symbol degrees higher than one, and in particular by demonstrating the ergodicity of the Ideal Soliton degree distribution, then by modeling and analyzing the ripple process, resulting in an unified model for both classical and doping-enhanced decoding. Based on that model, we analyze the collection cost of the presented decoding strategies, when the starting $\omega(d)$ is Ideal Soliton.

6.5.2 Symbol Degree Evolution

In this subsection, we focus on the evolution of symbol degrees higher than one (unreleased symbols), and then analyze ripple evolution separately in the next subsection. The analysis of the evolution of unreleased output symbols is the same for both classical BP decoding case (without doping), and the doped BP decoding. We now present the model of the doping (decoding) process through the column degree distribution at each decoding/doping round. We model the $(\ell+1)$th step of the decoding/doping process by selecting a row uniformly at random from the set of $(k-\ell)$ rows in the current $(k-\ell)\times k_s$ decoding matrix $\mathbf{S}_\ell$, and removing it from the matrix $\mathbf{S}_\ell$ to create the matrix $\mathbf{S}_{\ell+1}$. After ℓ rounds or, equivalently, when there are $k-\ell$ rows in the decoding matrix, the number of ones in a column of $\mathbf{S}_\ell$ is denoted by $A_{k-\ell}$. The probability that the column of $\mathbf{S}_{\ell+1}$ is of degree d (when its length is $k-\ell-1,\ \ell\in\{1,\cdots,k-3\}$) is described iteratively

$$P\left(A_{k-\ell-1}=d\right)=P\left(A_{k-\ell}=d\right)\left(1-\frac{d}{k-\ell}\right)+P\left(A_{k-\ell}=d+1\right)\frac{d+1}{k-\ell} \tag{6.5}$$

for $2\leq d<k-\ell$, and $P\left(A_{k-\ell-1}=k-\ell\right)=0$. Here, the first term corresponds to the probability that the degree of the column in the previous decoding round (i.e. for the matrix $\mathbf{S}_\ell$) was d and that the random row removed in the $(\ell+1)$th step did not contribute to the linear combination represented by that column. Similarly, the second term corresponds to the probability that the degree of the column in the previous decoding round was $d+1$ and that the row removed in the $(\ell+1)$th step contributed to the linear combination represented by that column.

Let the starting distribution of the column degrees (for the decoding matrix $\mathbf{S_0} = [s_{ij}]_{k\times k_s}$) be Ideal Soliton, as defined in (6.2). By construction, for $l = 0$, $P(A_k = d) = \rho(d)$, which, together with (6.5), completely defines the dynamics of the doping process when the Fountain code is based on the Ideal Soliton. After rearanging and canceling appropriate terms, we obtain, for $d \geq 2$,

$$P(A_{k-l} = d) \quad = \quad \begin{cases} \frac{k-l}{k}\rho(d) & d = 2, \cdots, k-l, \\ 0 & d > k-\ell. \end{cases} \tag{6.6}$$

We assume that $k_s \approx k$ as, by design, we desire to have the set of upfront collected symbols k_s as small as the set of source symbols. The probability of degree-d symbols among unreleased symbols $n_u^{(\ell)} = k_s - \ell$ can be approximated with

$$\frac{P(A_{k-\ell} = d)\, k_s}{k_s - \ell} \approx \frac{P(A_{k-\ell} = d)\, k}{k-\ell}. \tag{6.7}$$

Hence, the probability distribution $\omega_\ell(d)$ of the unreleased output node degrees at any time ℓ remains the Ideal Soliton

$$\omega_\ell(d) = \frac{k}{k-\ell} P(A_{k-\ell} = d) = \rho(d) \quad \text{for } d = 2, \cdots, k-\ell. \tag{6.8}$$

6.5.3 Doped Ripple Evolution: Random Walk Model

There exist comprehensive and thorough analytical models for the ripple evolution, characterizing the decoding of LT codes [78, 79]. However, their comprehensive nature results in complex models that are difficult to evaluate. For describing the dynamics of a doped decoder, we consider a simpler model, which attempts to capture the ripple evolution for the Ideal Soliton. Figure 6.7 and the code symbol degree evolution analysis illustrate how the Ideal Soliton distribution maintains its shape with decoding/doping. This fact, which results in a tractable ripple analysis and, more importantly, in an outstanding performance as illustrated in the last subsection, is our main motivator for selecting Ideal Soliton Fountain codes for our doping scheme. We study the number of symbols decoded between two dopings and, consequently, characterize the sequence of *interdoping yields*. The time at which the ith doping occurs (or,

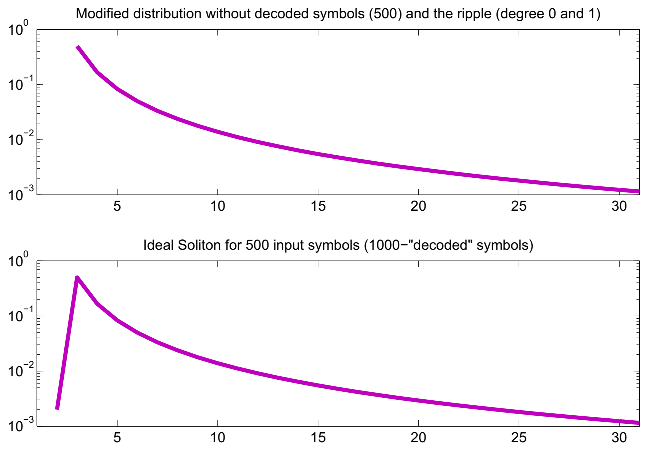


Figure 6.7: Density Evolution of IS distribution due to uniform doping. First graph is the distribution of the output symbols after $m = 500$ decodings, for $k = 1000$, second is the IS with support set $\{1, \cdots, (1000 - m)\}$ as if we are *starting* with the matrix of the same size as the doped matrix.

equivalently, the decoding stalls for the ith time) is a random variable T_i, and so is the inter-doping yield $Y_i = T_i - T_{i-1}$. Our goal is to obtain the expected number of times the doping will occur by studying the ripple evolution. This goal is closely related to (a generalization of) the traditional studies of the fountain code decoding which attempt to determine the number of collected symbols k_s required for the decoding to be achieved without a single doping iteration, i.e., when $T_1 \geq k$.

Let the number of upfront collected coded symbols be $k_s = k\,(1+\delta)$, where δ is a small positive value. At time ℓ, the total number of decoded and doped symbols is ℓ, and the number of (unreleased) output symbols is

$$n = k_s - \ell = \lambda_\ell^\delta\,(k-\ell)\,. \tag{6.9}$$

Here,

$$\lambda_\ell^\delta = 1 + \frac{k}{k-\ell}\delta \tag{6.10}$$

is an increasing function of ℓ. The unreleased output symbol degree distribution polynomial at time ℓ is $\Omega_\ell(x) = \sum \Omega_{d,\ell} x^d$, where $d = 2, \cdots, k-\ell$, and $\Omega_{d,\ell} = \omega_\ell(d)$. In order to describe the ripple process evolution, in the following we first characterize the ripple increment when ℓ corresponds to the decoding and, next, when it corresponds to a doping iteration.

Each decoding iteration processes a random symbol of degree-one from the ripple. Since the encoded symbols are constructed by independently combining random input symbols, we can assume that the input symbol covered by the degree-one symbol is selected uniformly at random from the set of undecoded symbols. Released output symbols are its coded symbol neighbors whose output degree is two. Releasing output symbols by processing a ripple symbol corresponds to performing, *in average*, $n_2 = n\Omega_{2,\ell}$ independent Bernoulli experiments with probability of success $p_2 = 2/(k-\ell)$. Hence, the number of released symbols at any decoding step ℓ is modeled by a discrete random variable $\Delta_\ell^{(\delta)}$ with Binomial distribution $\mathbf{B}\left(n\Omega_{2,\ell}, 2/(k-\ell)\right)$, which for large n can be approximated with a (truncated) Poisson distribution of intensity $2\Omega_{2,\ell}\lambda_\ell^{(\delta)}$

$$\begin{aligned}\Pr\left\{\Delta_\ell^{(\delta)} = r\right\} &= \binom{n_2}{r}(p_2)^r(1-p_2)^{n_2-r} \\ &\geq \frac{(n_2)^r}{r!}(p_2)^r(1-p_2)^{n_2-r} \\ &\approx \frac{(2\Omega_{2,\ell}\lambda_\ell^{(\delta)})^r}{r!}e^{-2\Omega_{2,\ell}\lambda_\ell^{(\delta)}}, r = 0,\cdots,n_2, \end{aligned} \tag{6.11}$$

where we have first applied the Stirling approximation to the Binomial coefficient and, also, assumed that the probabilities in (6.11) can be neglected unless n_2 is much larger than r. According to (6.8), the fraction of degree-two output symbols for Ideal Soliton based Fountain code is expected to be $n_2/n \approx \Omega_{2,\ell} = \rho(2) = 1/2$, for any decoding iteration ℓ. Hence,

$$\Pr\left\{\Delta_\ell^{(\delta)} = r\right\} = \eta(r) = \frac{\left(\lambda_\ell^{(\delta)}\right)^r e^{-\lambda_\ell^{(\delta)}}}{r!}, \quad r = 0,\cdots,n/2, \tag{6.12}$$

or, equivalently,

$$\Delta_\ell^{(\delta)} \sim \wp\left(\lambda_\ell^{(\delta)}\right), \tag{6.13}$$

where $\wp(\cdot)$ denotes Poisson distribution. For each decoding iteration, one symbol is taken from the ripple and $\Delta_\ell^{(\delta)}$ symbols are added, so that the increments of the ripple process can be described by random variables

$$X_\ell = \Delta_\ell^{(\delta)} - 1, \tag{6.14}$$

with the probability distribution $\eta(r+1)$ (for $X_\ell = r$) characterized by the generating polynomial

$$I(x) = \sum_{d=0}^{n/2} \eta(d) x^{d-1} \tag{6.15}$$

and an expected value $\lambda_\ell^{(\delta)} - 1$. Next we describe the ripple increment for the doping iteration, where a carefully selected input symbol is revealed at time $T_i = t_i$ when the ripple is empty (random degree-two doping). The number of degree-two output symbols at time $T_i = t_i$ is $n_2 = \rho(2)n = n/2$, where, $n = \lambda_{t_i}^{(\delta)}(k - t_i)$.

Degree-two doping selects uniformly at random a row in the decoding matrix $\mathbf{S_{t_i}}$ that has one or more non-zero elements in columns of degree two. This is equivalent to randomly selecting a column of degree two to be released, and restarting the ripple (i.e., same as decoding) with any of its two input symbols from the decoding matrix whose number of degree-two columns is now $n_2 - 1 \approx n_2$, for large n_2. Hence, the doping ripple increment can be described by unit increase in addition to an increase equivalent to the one obtained through decoding but *without* the ripple decrement of 1. That is, statistically, the doping ripple increment $X_{t_i}^D$ is a random variable described by

$$I^D(x) = \sum \eta(d) x^{d+1}, \tag{6.16}$$

corresponding to the shifted distribution $\eta(r-1)$ for $X_{t_i}^D = r$.

Now if, for the doping instant $t = t_{i-1}$, we define $X_{t_{i-1}} = X_{t_{i-1}}^D - 2$, the ripple size for $t \in [t_{i-1},\ t_i]$ can be described in a unified manner with $S_{t,i} + 2$ where

$$S_{t,i} = \sum_{j=t_{i-1}}^{t} X_j \tag{6.17}$$

is a random walk modeling the ripple evolution. Note that the ripple increments X_ℓ are not IID random variables, since the intensity of $\eta(d)$ changes with each iteration ℓ. However, for analytical tractability, we study the interdoping time using the random walk model in (6.17), by assuming that $\lambda^{(\delta)}$ changes from doping to doping, but remains constant within the interdoping interval. Under this assumption, the ripple size $S_{t,i}+2$ is a partial sum of IID random variables X_j, of the expected value $\lambda^{(\delta)}_{t_{i-1}}-1$. Note that, when $\delta=0$, i.e. when $k_s=k$, $S_{t,i}$ is a zero mean random walk. In this special case, we treat the doping-enhanced BP process as an approximate renewal process, where the process starts all over after each doping. Modeling and analyzing this particular case is much easier, resulting in a closed-form expression for the expected number of dopings. We later refer to this case to provide some intuition. The expected interdoping yield is the expected time it takes for the ripple random walk $S_{t,i}+2$ to become zero. Using random walk terminology, we are interested in the statistics of the random-walk stopping time. The stopping time is the time at which the decoding process stalls, counting from the previous doping time, where the first decoding round starts with the 0th doping which occurs at $T_0=0$. Hence, the i-th stopping time (doping) T_i is defined as

$$T_i \quad = \quad \min\left\{\min\left\{t_i : S_{t,i}+2\leq 0\right\}, k\right\}. \tag{6.18}$$

We study the Markov Chain model of the random walk $S_{t,i}$. Each possible value of the random walk represents a state of the Markov Chain *(MC)* described by the probability transition matrix $\mathbf{P}_i$. State $v, v\in\{1,\cdots,k\}$ corresponds to the ripple of size $v-1$. State 1 is the trapping state, with the (auto)transition probability $\mathbf{P}_{i,11}=1$ and models the stopped random walk. Hence, based on (6.12), we have the state transition probabilities

$$\begin{aligned}
\mathbf{P}_{i,v(v+b)} &= \eta(1+b),\ v=2,\cdots,k, b=-1,\cdots,\min\left(\lceil n/2\rceil, k-v\right)\\
\mathbf{P}_{i,11} &= 1\\
\mathbf{P}_{i,vw} &= 0, \quad \text{otherwise},
\end{aligned} \tag{6.19}$$

resulting in a transition probability matrix of the following almost Toeplitz form

$$\mathbf{P}_i \;=\; \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \eta(0) & \eta(1) & \eta(2) & \cdots & 0 \\ 0 & \eta(0) & \eta(1) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \eta(0) & \eta(1) \end{bmatrix}_{k\times k} , \qquad (6.20)$$

where $\eta(\cdot)$ represents a pdf from a family of Poisson distributions $\wp\left(\lambda_{t_i}^{(\delta)}\right), t_i \in [0, \cdots, k)$. The start of the decoding process is modeled by the MC being in the initial state $v = 3$ (equivalent to the ripple of size two). Based on that, the probability of being in the trapping state, while at time $t > T_i$, is

$$p_t^{(T_i)} = [0\,0\,1\,0\cdots 0]\,\mathbf{P}_i^{(t-T_i)}\,[1\,0\,0\cdots 0]^T . \qquad (6.21)$$

Hence, the probability of entering the trapping state at time t is

$$\begin{aligned} p^{T_i}(u) &= p_{T_i+u}^{(T_i)} - p_{T_i+u-1}^{(T_i)} \\ &= [0\,0\,1\,0\cdots 0]\left(\mathbf{P}_i^u - \mathbf{P}_i^{(u-1)}\right)[1\,0\,0\cdots 0]^T , \end{aligned} \qquad (6.22)$$

where $u = t - T_i$. $\{T_i\}$ is a sequence of stopping-time random variables where index i identifies a doping round. $Y_i = T_i - T_{i-1}, i > 1$ is a stopping time interval of a random walk of (truncated) Poisson IID random variables of intensity $\lambda_{T_{i-1}}^{(\delta)} = 1 + \delta\frac{k}{k-T_{i-1}}$, and can be evaluated using the following recursive probability expression

$$\begin{aligned} \Pr\{Y_i = 0\} &= 0 \qquad (6.23) \\ \Pr\{Y_i = 1\} &= 0 \\ \Pr\{Y_i = t+1\} &= \eta(0)\left(\aleph^{(t)}(t-1) - \sum_{i=1}^{t-1}\Pr\{Y_i = t-i\}\,\aleph^{(i)}(1+i)\right) \qquad 1 \le t < k, \end{aligned}$$

obtained from (6.22) after a series of matrix transformations. Here, $\eta(0)$ is a Poisson pdf of intensity $\lambda_{T_{i-1}}^{(\delta)}$ evaluated at 0, and $\aleph^{(s)}(d)$ is the s-tuple convolution of $\eta(\cdot)$ evaluated at d, resulting in a Poisson pdf of intensity $s\lambda_{T_{i-1}}^{(\delta)}$ evaluated at d. The complete derivation of (6.23) is given in the Appendix. Note that the intensity $s\lambda_{T_{i-1}}^{(\delta)}$ is, in general, a random variable and that the sequence of doping times T_i is a Markov chain. Hence, the number of decoded symbols after hth doping, a partial sum

$$D_h = \sum_{i=1}^{h} Y_i \tag{6.24}$$

of interdoping yields, is a Markov-modulated random walk. The expected number of dopings sufficient for complete decoding is the stopping time of the random walk D_h, where the stopping threshold is $k - u_k^\delta$. Here, based on the coupon collection model, u_k^δ is the expected number of uncovered symbols (which, necessarily, have to be doped) when k_s coded symbols are collected

$$u_k^\delta = k\left(1 - \frac{1}{k}\right)^{[k(1+\delta)\log k]} \approx k\mathbf{e}^{-(1+\delta)\log k}. \tag{6.25}$$

The total number of dopings is the stopping time random variable

$$D \quad = \quad \min\left\{h : D_h + u_k^\delta \geq k\right\}. \tag{6.26}$$

Our model can further be simplified by replacing T_{i-1} with $l_i = \sum_{t=1}^{i-1} E\left[Y_t | T_{t-1} = l_t\right]$ in the intensity $\lambda_{T_{i-1}}^{(\delta)}$ (6.23) and thus allowing for a direct recursive computation in (6.23). Hence,

$$E\left[Y_i | T_{i-1} = l_i\right] \quad \approx \textstyle\sum_{t=1}^{k-l_i} t\text{Pr}\left\{Y_i = t\right\} + \left(1 - \sum_{t=1}^{k-l_i} \text{Pr}\left\{Y_i = t\right\}\right)(k - l_i). \tag{6.27}$$

Furthermore, we can approximate D_h with the sum of expected values $l_{h+1} = \sum_{i=1}^{h} E\left[Y_i | T_{i-1} = l_i\right]$ and use an algorithm in Figure 6.8 (based on (6.26)) to calculate expected number of dopings. In special case when $\delta = 0$, further simplifying assumptions lead to the approximation that all interdoping yields are described by a single random variable Y whose pdf is given by the following recursive expression, based on (6.23),

$$\text{Pr}\left\{Y = t+1\right\} = \eta(0)\left(\wp^{(t)}(t-1) - \sum_{i=1}^{t-1} \text{Pr}\left\{t - i\right\}\wp^{(i)}(1+i)\right), \tag{6.28}$$

where $\wp^{(s)}(d)$ denotes Poisson distribution of intensity s, evaluated at d, and $t \in [0,\ k-1]$.

```
**Initialization:**
    $l_i = 0, D = 0$
**For** $(i = 1, D < k, i++)$
    Calculate $\lambda^{(\delta)}(l_i)$
    Using (6.23), calculate Pr $\{Y_i = t\}$ for $t \leq k - l_i$
    Using (6.27), calculate $E[Y_i]$
    $D = D + E[Y_i]$
    $l_i = D$
$k_d = i, p_d = 100k_d/k$
```

Figure 6.8: Calculation of the expected doping percentage p_d based on the number of upfront collected symbols

The range of t varies from doping to doping, i.e. if $T_{i-1} = l_i$, then Y_i would have support $t \in [l_i,\ k-1]$, and, hence, this single variable approximation is accurate for the case when both the ripple size is small and when $l_i \ll k$. We now approximate the expected value of the interdoping yield Y as

$$E[Y] \approx \sum_{t=1}^{k} t\mathrm{Pr}\{Y = t\} - \left(1 - \sum_{t=1}^{k} \mathrm{Pr}\{Y = t\}\right) k. \tag{6.29}$$

Now, the doping process D_h is a renewal process, and thus, the Wald Equality [50] implies that the mean stopping time is $E[D] = k/E[Y]$.

6.6 Comparative Cost Analysis

The summary of the proposed approach to dissemination, storage, and collection with doping, based on IS combining for storage, and a random degree-two doping for collection strategy, is given in Figure 6.9. We here analyze the performance of this approach in terms of data collection cost. The cost of the upfront collection from the nearby nodes in the super squad $1 + (s-1)/4$ is significantly smaller than the collection cost when the packets are polled from their original source relays, which is in average $k/4$. Nevertheless, in this section, we show that the number of doped packets k_d will be sufficiently smaller than the residual number of

```
Dissemination and Storage:
    degree-one/two dissemination of $k$ source packets; each storage node stores a random linear
        combination of $d$ disseminated packets; $d$ is drawn from IS $\rho(d)$.
Upfront collection:
    IDC collects $k_s$ encoded packets from $s$ closest storage squads.
Belief propagation decoding and doping-collection:
    $l = 0$: number of processed source packets
    $k_{r,l}$: number of packets in the ripple
    $k_d = 0$: number of doped packets.
For $(l = 0, l \leq k, l++)$
    while $k_{r,l} = 0$
        Collect(from the source relay) and dope the decoder with a source packet contributing to a randomly
            selected degree-two (or larger) output packet.
        $k_d++; l++;$
    endwhile
    Process a symbol from the ripple; $k_{r,l}--$;
endfor
```

Figure 6.9: Proposed dissemination, storage, and doping collection

undecoded symbols when the belief propagation process first stalls, so that their collection cost is offset, and the overall collection cost is reduced relative to the original strategy. We quantify the performance of the decoding process through the doping ratio k_d/k. Figure 6.10 illustrates the dramatic overhead $(k_T - k)/k$ reduction when employing doping with an IS distribution relative to the overhead of RS encoding without doping. Figure 6.11 demonstrates that RS with doping performs markedly worse than IS encoding. And, in particular, it illustrates that IS with doping demonstrates a very low variance, which is surprisingly different from the results without doping. Figure 6.13 illustrates the importance of considering coverage redundancy when selecting storage/ collection strategy: in the case of degree-two dissemination, when the size of the supersquad s increases the fountain code strategy improves (in terms of a reduced doping k_d/k required for decoding) due to an increase in mixing. This dependency is not present in the case of degree-one dissemination. Figure 6.12 gives the corresponding required

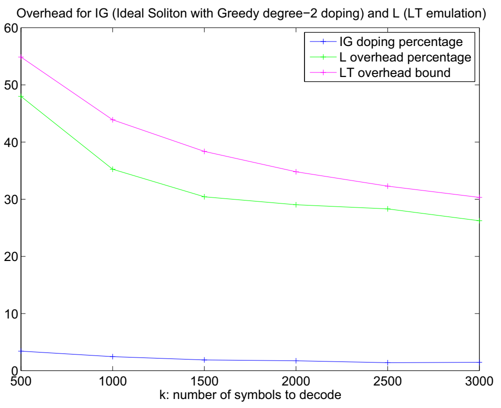


Figure 6.10: Overhead (doping) percentage: we define k_T as the number of symbols collected in both collection phases, and the collection overhead ratio as $(k_T - k)/k$, which allows us to compare the overhead for the LT decoding of k original symbols and IG, the degree-two doped belief-propagation decoding of k coded symbols with IS degree distribution.

doping k_d/k as a function k for a fixed squad size $h = 200$. The *cost minimization problem* for any encoding scheme with (and without) doping is described as follows. Let, the pair (k_s, k_d) be the feasible number of encoded and doped packets when sufficient for decoding the original k packets. The per-source packet collection cost for this pair is

$$c_T(h) = [c_s(h)k_s + c_d k_d]/k \tag{6.30}$$

where $c_s(h) = 1 + (s(h) + 1)/4$ is the average collection cost from the supersquad of size $s(h) = \lceil k_s/h \rceil$ and $c_d = \lceil k/4 \rceil$ is the average collection doping cost when polling doped packets from the original source relays. Examples of (k_s, k_d) pairs are $(0, k)$ for the pure polling mechanism with cost $c_T(h) = c_d = \lceil k/4 \rceil$ and $(k_s = k + \sqrt(k)\log^2(k/\delta), 0)$ in average for degree-one dissemination and RS fountain encoding with average per-packet cost $c_T = c_s(h)k_s/k$. For any given encoding mechanism and the set of feasible pairs (k_s, k_d), the minimum per-packet collection cost is $c_{min}(h) = \min_{(k_s,k_d)} c_T(h)$. The effect on the doping percentage of increasing the number of upfront collected symbols k_s above k (described by our general model of interdoping times) is illustrated in Figure 6.14. Figure 6.15 illustrates per-packet collection cost above minimum, based on (6.30), as a function of the number of packets

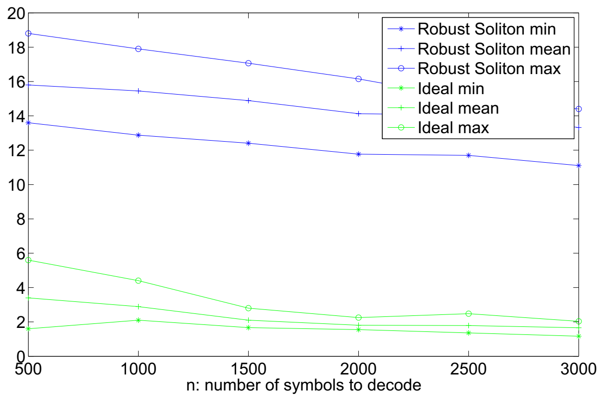


Figure 6.11: Doping percentage with IS degree distribution vs RS.

$(k_s - k)/k$ collected from the supersquad in excess of k, for different values of coverage redundancy h, and IS encoding. For the range of coverage redundancies that may be of practical value (up to 50), the minimum collection cost is obtained for $k_{s,min}/k \in (1, 1.05)$. Figure 6.16 illustrates the per-packet cost $c_T(h)/(k/4)$ normalized to the reference polling cost as a function of $\lambda_s/\lambda = 1/h$, the relative density of source nodes for a network with $k = 2000$ source packets. Four strategies are included all based on degree-1 packet dissemination: reference polling, degree-1 coupon collection, RS with no doping, and the IS encoding with a feasible doping pair (k_s, k_d).

In conclusion, in this paper we showed that, for the circular squad network, the total collection cost could be reduced by applying a packet combining degree distribution that is congruous to doping, applying a good doping mechanism, and by balancing the cost of upfront collection and doping, given coverage redundancy factor.

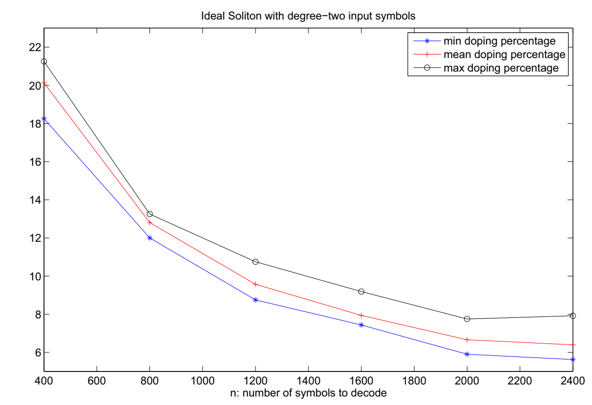


Figure 6.12: The encoding process emulates supersquads with fixed squad size h and the degree-two input symbols overheard within the superquad: the resulting doping percentage for IS degree distribution of stored code symbols.

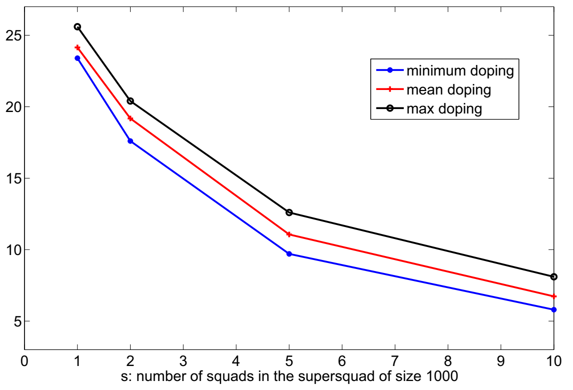


Figure 6.13: For a fixed number of upfront collected symbols $k_s = 1000$, encoded by degree-two IS method, the squad size (node density) is changed, so that the supersquad contains $1, 2, 5,$ and 10 squads. The more squads there are, the more intense is the data mixing, decreasing the probability of non-covered original symbols.

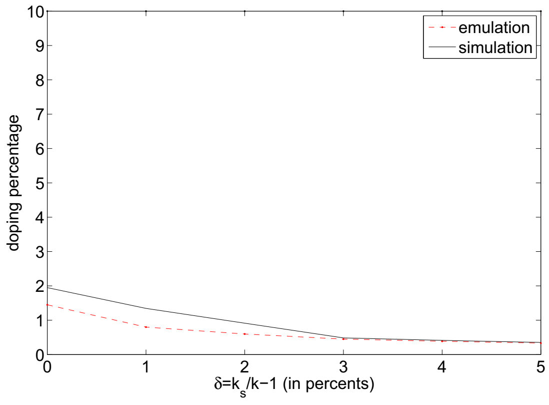


Figure 6.14: Doping percentage for different values of $\delta = k_s/k - 1$. Emulation results are obtained based on our analytical model and algorithm in Figure 6.8

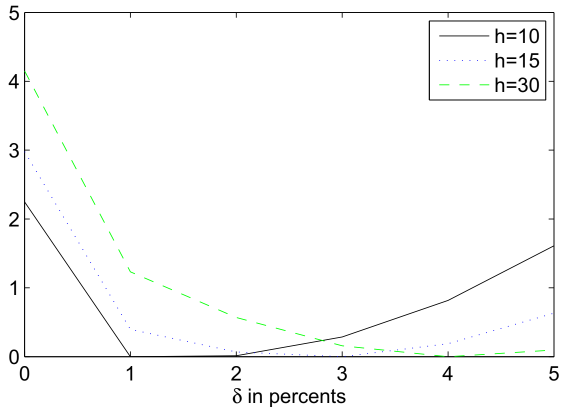


Figure 6.15: Collection delay (hop count) above minimum per input symbol for different values of coverage redundancy h as a function of δ. Note that there is an optimal δ for each h in which the delay is minimized: for $h = 10$ δ is one percent, for $h = 15$ it is 3% percent, for $h = 30$ $\delta = 4\%$

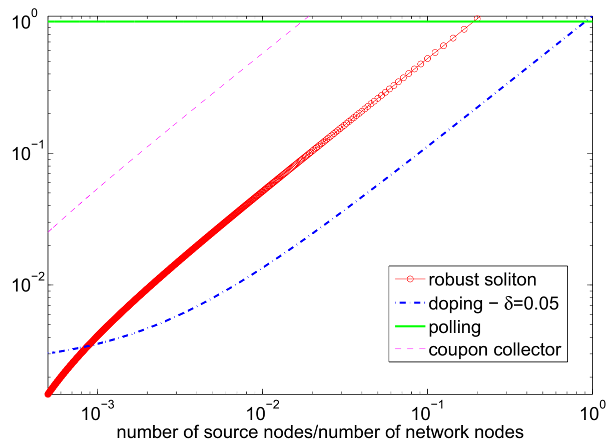


Figure 6.16: Collection Delay for various collection techniques, normalized with respect to the polling cost

Appendix A

Outage Constraint Derivations

Given (2.23), we expand (2.24) as

$$\begin{aligned}\kappa_j^{(\eta)} &= \sum_{i=1}^{m} \text{Pr}\left\{\left(\hat{L}_{\eta-1} = h_i, A_{\eta-1}\right), \hat{L}_\eta = h_j, Z_\eta > 0\right\} \\ &= \sum_{i=1}^{m} \text{Pr}\left\{\hat{L}_\eta = h_j, Z_\eta > 0 | \hat{L}_{\eta-1} = h_i, A_{\eta-1}\right\} \\ &\text{Pr}\left\{\hat{L}_{\eta-1} = h_i, A_{\eta-1}\right\} \\ &= \sum_{i=1}^{m} \text{Pr}\left\{\hat{L}_\eta = h_j, Z_\eta > 0 | \hat{L}_{\eta-1} = h_i\right\} \kappa_i^{(\eta-1)}. \end{aligned} \tag{A.1}$$

Let us now expand $\text{Pr}\left\{\hat{L}_\eta = h_j, Z_\eta > 0 | \hat{L}_{\eta-1} = h_i\right\}$

$$\begin{aligned} &= \text{Pr}\left\{Z_\eta > 0 | \hat{L}_\eta = h_j, \hat{L}_{\eta-1} = h_i\right\} \text{Pr}\left\{\hat{L}_\eta = h_j | \hat{L}_{\eta-1} = h_i\right\} \\ &= \text{Pr}\left\{Z_\eta > 0 | \hat{L}_\eta = h_j\right\} \text{Pr}\left\{\hat{L}_\eta = h_j | \hat{L}_{\eta-1} = h_i\right\}, \end{aligned} \tag{A.2}$$

where in the first term, invoking Markovity, we drop the second condition since Z_η does not depend on $\hat{L}_{\eta-1}$, given $\hat{L}_\eta$. We recognize the term $\text{Pr}\left\{\hat{L}_\eta = h_j | \hat{L}_{\eta-1} = h_i\right\}$ as P_{ij}. We also note that given the spoke is in state $\hat{L}_k = h_j$ at time η, thus inducing a crescent C_k of area c_j, the crescent C_k is empty with probability

$$\overline{e_j} = \text{Pr}\left\{Z_k > 0 | L_k = h_j\right\} = 1 - \exp(-\lambda c_j). \tag{A.3}$$

It follows from (A.1), (A.2) and (A.3) that $\kappa_j^{(\eta)} = \sum_{i=1}^{m} \overline{e_j} P_{ij} \kappa_i^{(\eta-1)}$, which, following the definition $\breve{P}_{ij} = P_{ij}\overline{e_j}$ in (2.26), results in (2.27) and (2.28), and eventually in the formalization of the Outage Constraint given by (2.31).

For the MC model with $m = 2$ states, performing singular value decomposition (SVD) of the two-state $\breve{\mathbf{P}}$ and applying it to (2.31) yields

$$\Pr\{D \leq \eta\} = U_1 \lambda_1^{\eta-1} + U_2 \lambda_2^{\eta-1} \tag{A.4}$$

where for $i = 1, 2$ the eigenvalues of $\breve{\mathbf{P}}$ are

$$\lambda_i = \frac{\breve{P}_{22}}{2}\left(1 + (-1)^{i+1}\sqrt{1 + \frac{4\breve{P}_{12}\breve{P}_{21}}{\breve{P}_{22}^2}}\right) \tag{A.5}$$

and the coefficients are

$$U_i = (-1)^i e_2 \frac{\breve{P}_{21} + \lambda_i}{\lambda_2 - \lambda_1}. \tag{A.6}$$

In the following, we provide closed form approximation of (A.4) for the non-uniform quantization model and $c_{22} = c_2 \gg c_1 = 1$, or equivalently $r \gg 1$ and q close to two (or δ close to zero).

Next, we introduce $\tilde{P}_{ij} = P_{ij}e_i$, which is convenient for the interpretation of the derived expressions. We observe that $\tilde{P}_{ij}$ is the probability of transitioning to state j from state i, and that the corresponding initial crescent C_k of area c_i is not empty. In addition, $\tilde{P}_{ie} = 1 - e_i$ is the probability that the crescent of area c_i is empty. Since $\breve{P}_{ij}\breve{P}_{ji} = \tilde{P}_{ij}\tilde{P}_{ji}$, and $\breve{P}_{ii} = \tilde{P}_{ii}$, the eigenvalues (A.5) have the same form expressed in terms of $\tilde{P}_{ij}$, Next, let us show that λ_1 can be approximated as follows

$$\begin{aligned} &\lambda_1 \approx 1 - \epsilon_o \quad \text{where} \\ &\epsilon_o = \frac{\tilde{P}_{21}\tilde{P}_{1e} + \tilde{P}_{2e}}{2 - \tilde{P}_{22}}. \end{aligned} \tag{A.7}$$

Starting from

$$\begin{aligned}
\lambda_1 &= 1-\left(\frac{2-\tilde{P}_{22}}{2}-\sqrt{\frac{\tilde{P}_{22}^2}{4}+\tilde{P}_{12}\tilde{P}_{21}}\right) && \text{(A.8)}\\
&= 1-\frac{2-\tilde{P}_{22}}{2}\left(1-\sqrt{\frac{\tilde{P}_{22}^2+4\tilde{P}_{12}\tilde{P}_{21}}{\left(2-\tilde{P}_{22}\right)^2}}\right) && \text{(A.9)}\\
&= 1-\epsilon, && \text{(A.10)}
\end{aligned}$$

and further simplifying ϵ,

$$\begin{aligned}
\epsilon &= \frac{2-\tilde{P}_{22}}{2}\left(1-\sqrt{\frac{\tilde{P}_{22}^2+4\left(1-\tilde{P}_{1e}\right)\left(1-\tilde{P}_{22}-\tilde{P}_{2e}\right)}{\left(2-\tilde{P}_{22}\right)^2}}\right) && \text{(A.11)}\\
&= \frac{2-\tilde{P}_{22}}{2}\left(1-\sqrt{\frac{\left(2-\tilde{P}_{22}\right)^2-4\left(\tilde{P}_{21}\tilde{P}_{1e}+\tilde{P}_{2e}\right)}{\left(2-\tilde{P}_{22}\right)^2}}\right) && \text{(A.12)}\\
&= \frac{2-\tilde{P}_{22}}{2}\left(1-\sqrt{1-\frac{4\left(\tilde{P}_{21}\tilde{P}_{1e}+\tilde{P}_{2e}\right)}{\left(2-\tilde{P}_{22}\right)^2}}\right), && \text{(A.13)}
\end{aligned}$$

we obtained an expression in which $\gamma = 4\frac{\tilde{P}_{21}\tilde{P}_{1e}+\tilde{P}_{2e}}{\left(2-\tilde{P}_{22}\right)^2}$ is a small value for the design in which we seek to minimize the probability of encountering an empty crescent. Now, we will expand the value of ϵ in (A.13) around this small value γ as follows

$$\epsilon = \frac{2-\tilde{P}_{22}}{2}\left(1-\left(1-\frac{1}{2}\gamma+R_2(\gamma)\right)\right). \tag{A.14}$$

Finally, including (A.14) in (A.10), we obtain the approximated formula (A.7), as follows

$$\begin{aligned}
\lambda_1 &= 1-\frac{2-\tilde{P}_{22}}{2}\left(1-\left(1-\frac{1}{2}\gamma+R_2(\gamma)\right)\right)\\
&\approx \frac{\tilde{P}_{22}}{2}+\frac{2-\tilde{P}_{22}}{2}\left(1-\frac{1}{2}\gamma\right)\\
&= 1-\frac{\tilde{P}_{21}\tilde{P}_{1e}+\tilde{P}_{2e}}{2-\tilde{P}_{22}}.
\end{aligned}$$

Note also that

$$\lambda_2 = \tilde{P}_{22} - \lambda_1. \tag{A.15}$$

We observe that λ_1 is close to one if ϵ_o is small, and ϵ_o should be small for the design in which we seek to make small the probability of encountering an empty crescent P_{ie}, as the numerator of ϵ_o in (A.7) represents the probability of encountering an empty crescent either in state 2 ($\tilde{P}_{2e}$), or after transitioning from state 2 to state 1 ($\tilde{P}_{21}\tilde{P}_{1e}$). Note that for $c2 \gg 1$, P_{2e} is small. We also have that

$$\begin{aligned}\tilde{P}_{22} &= 1 - \left(\tilde{P}_{21} + \tilde{P}_{2e}\right)\\ &= 1 - \frac{e_2}{c_2} - (1 - e_2) = \frac{(c_2 - 1)\left(1 - e^{-c_2}\right)}{c_2} \approx 1.\end{aligned} \tag{A.16}$$

Note that here we assumed that $c_{21} = c_{12} = c_1 = 1$ for large enough r, where the details of derivation follow soon. Hence, since $1 > \lambda_1 > \tilde{P}_{22}$, then, from (A.15) and (A.16), λ_2 is a small negative value.

From (A.6), since $e_2\tilde{P}_{21}$ is close to zero, we see that the coefficient U_1 is close to one, while U_2 is close to zero. Hence, we can neglect the second term in (A.4). We see now that the larger eigenvalue λ_1 describes the rate at which outage probability increases with the number of hops, while the negative eigenvalue λ_2 describes the oscillatory, self-recovery mechanism depicted in Figure 2.4.

Thus, following (A.7), the equality (A.4) can be now approximated with

$$\Pr\{D \leq \eta\} \approx (1 - \epsilon_o)^{\eta - 1}. \tag{A.17}$$

Finally, replacing (A.17) in (2.22), we obtain

$$p \geq 1 - (1 - \epsilon_o)^{\eta - 1}. \tag{A.18}$$

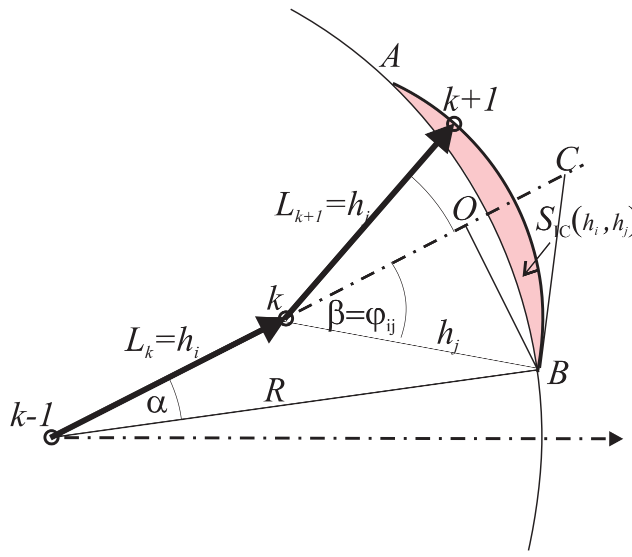


Figure A.1: Given $L_k = h_i$ and $L_{k+1} = h_j$, the angular hop displacement Φ_{k+1} is constrained to the interval $-\beta \leq \Phi_{k+1} \leq \beta$ where the maximum angular displacement at hop $k+1$ is $\beta = \beta(h_i, h_j)$. The shaded area denotes the interior crescent of area $S_{\text{IC}}(h_i, h_j)$.

Also, as from (A.16) $\tilde{P}_{22} \approx 1$, and $\tilde{P}_{2e}$ is small enough to be safely ignored,

$$\epsilon_o = \frac{\tilde{P}_{21}\tilde{P}_{1e} + \tilde{P}_{2e}}{2 - \tilde{P}_{22}} = \frac{c_{21}e^{-1}}{c_2} = \frac{e^{-1}}{c_2}, \tag{A.19}$$

since $\tilde{P}_{1e} = e^{-1}$, and $\tilde{P}_{21} = \frac{c_{21}}{c_2} = \frac{1}{c_2}$.

Next, we explain the approximations used in our asymptotic analysis for the crescent areas, leading to the expression for c_2. For any interior crescent, c_{ij}, the width of the crescent is $\Delta_{ij} = h_i + h_j - R = \Delta_{ji}$. From Figure A.1 we approximate the interior crescent angle $\beta(h_i, h_j)$ with the value of its sinus function, calculated using similarity of the triangles CBk and BOk:

$$\begin{aligned}\beta(h_i, h_j) = \varphi_{ij} &= \frac{\overline{OB}}{h_j} \\ &\approx \frac{\overline{BC}}{\sqrt{h_j^2 - \overline{OB}^2} + \Delta_{ij}},\end{aligned} \tag{A.20}$$

from which we first approximate $\overline{OB}^2 = 2\Delta_{ij}h_j$, where Δ_{ij}^2 is considered small and neglected, and then, from (A.20),

$$\varphi_{ij}^2 = 2\frac{\Delta_{ij}}{h_j}. \tag{A.21}$$

We approximate the crescent area c_{ij} with the area of the circular segment (of subtending angle $2\beta(h_i, h_j)$), which is the area of circular sector less the area of the triangle $\angle AkB$

$$\begin{aligned} c_{ij} &= h_j^2 \varphi ij - (h_j - \Delta_{ij})\sqrt{2\Delta_{ij}h_j} \\ &= \Delta_{ij}\sqrt{2\Delta_{ij}h_j} \\ &= \sqrt{h_j}\sqrt{2}\Delta_{ij}^{3/2}. \end{aligned} \tag{A.22}$$

Since for uniformly quantized m-state MC

$$\begin{aligned} \frac{\Delta_{ij}}{h_j} &= \frac{h_i + h_j - R}{h_j} \\ &= 1 + \frac{q\,(m-i) + 2i - m - qm}{q\,(m-j) + 2j - m} \\ &= 1 + \frac{i\,(2-q) - m}{q\,(m-j) + 2j - m}, \end{aligned} \tag{A.23}$$

we have

$$\begin{aligned} c_{ij} &= h_j^2\sqrt{2}\frac{\Delta_{ij}}{h_j}^{3/2} \\ &= \sqrt{2}r^2\frac{(q\,(m-j) + 2j - m)^2}{m^2}\left(1 + \frac{i\,(2-q) - m}{q\,(m-j) + 2j - m}\right)^{3/2}. \end{aligned} \tag{A.24}$$

Hence,

$$\begin{aligned} c_2 = c_{mm} &= \sqrt{2}r^2\left(1 + \frac{m\,(2-q) - m}{m}\right)^{3/2} \\ &= r^2 f(q), \end{aligned} \tag{A.25}$$

where

$$f(q) = \sqrt{2}\,(2-q)^{3/2}. \tag{A.26}$$

is the area of largest crescent c_2 formed for unit range r.

Appendix B

Wobbliness Constraint Derivations

For $m = 2$ and $P_{11} = 0$, the matrix we introduced in (2.18) can be expressed as

$$\mathbf{\Gamma} = \begin{bmatrix} 0 & \hbar\left(\varphi_{12}\omega\right) \\ \hbar\left(\varphi_{21}\omega\right)\frac{c_1}{c_2} & \hbar\left(\varphi_{22}\omega\right)\frac{c_2-c_{21}}{c_2}, \end{bmatrix} \tag{B.1}$$

where the definition of $\hbar\left(a\omega\right)$ is given in (2.16). Some algebra will verify that its largest eigenvalue has the following value

$$\begin{aligned}\sigma\left(\omega\right) = & \frac{\hbar\left(\varphi_{22}\omega\right)P_{22}}{2} \\ & + \frac{\sqrt{\left[\hbar\left(\varphi_{22}\omega\right)P_{22}\right]^2 + 4\hbar^2\left(\varphi_{12}\omega\right)P_{12}P_{21}}}{2},\end{aligned} \tag{B.2}$$

and that a scaled eigenvector $\nu(\omega)$ has components

$$\nu_1\left(\omega\right) = \frac{\Gamma_{12}\left(\omega\right)}{\sigma\left(\omega\right)}, \qquad \nu_2(\omega) = 1. \tag{B.3}$$

Let us first highlight a couple of facts we have used to find values of second derivatives of terms in (2.40).

$$\lim_{\omega\to 0} \hbar\left(a\omega\right) = 1 \tag{B.4}$$

$$\frac{\partial}{\partial\,\omega}\hbar\left(a\omega\right) = \hbar'\left(a\omega\right) = \frac{coshա a\omega}{\omega} - \frac{\hbar\left(a\omega\right)}{\omega} \tag{B.5}$$

$$\lim_{\omega\to 0}\frac{\partial}{\partial\,\omega}\hbar\left(a\omega\right) = 0 \tag{B.6}$$

$$\frac{\partial^2}{\partial\,\omega^2}\hbar\left(a\omega\right) = \hbar''\left(a\omega\right) = a^2\hbar\left(a\omega\right) - \frac{\cosh a\omega + \omega\frac{\partial\,\hbar(a\omega)}{\partial\,\omega} - \hbar\left(a\omega\right)}{\omega^2} \tag{B.7}$$

$$\lim_{\omega\to 0}\frac{\partial^2}{\partial\,\omega^2}\hbar\left(a\omega\right) = \frac{a^2}{3} \tag{B.8}$$

$$\begin{aligned} u_1\left(\omega\right) &= P_{22}\hbar\left(\varphi_{22}\omega\right) - P_{11}\hbar\left(\varphi_{11}\omega\right) \\ &\quad + \sqrt{\left(P_{22}\hbar\left(\varphi_{22}\omega\right) - P_{11}\hbar\left(\varphi_{11}\omega\right)\right)^2 + 4\left(\hbar\left(\varphi_{12}\omega\right)\right)^2 P_{12}P_{21}}. \end{aligned} \tag{B.9}$$

Based on (B.3), the right eigenvector elements evaluated for $\omega = 0$ are:

$$\left.\nu_1\left(\omega\right)\right|_{\omega=0} = K \tag{B.10}$$

$$\left.\nu_2\left(\omega\right)\right|_{\omega=0} = K. \tag{B.11}$$

The first derivative of the eigenvalue evaluated for $\omega = 0$ gives

$$\begin{aligned} \left.\frac{\partial}{\partial\,\omega}\sigma\left(\omega\right)\right|_{\omega=0} &= \frac{1}{2}\left(P_{11}\,\hbar'\left(\varphi_{11}\omega\right) + P_{22}\,\hbar'\left(\varphi_{22}\omega\right)\right) + \\ &\quad \frac{\sigma\sigma(\omega)}{2\sqrt{\left(\hbar\left(\varphi_{22}\omega\right)P_{22} - \hbar\left(\varphi_{11}\omega\right)P_{11}\right)^2 + 4\hbar\left(\varphi_{12}\omega\right)^2 P_{12}P_{21}}} \\ &= 0, \end{aligned} \tag{B.12}$$

where

$$\begin{aligned} \sigma\sigma(\omega) &= \left(P_{22}\hbar\left(\varphi_{22}\omega\right) - P_{11}\hbar\left(\varphi_{11}\omega\right)\right)\left(P_{22}\,\hbar'\left(\varphi_{22}\omega\right) - P_{11}\,\hbar'\left(\varphi_{11}\omega\right)\right) \\ &\quad + 4P_{12}P_{21}\hbar\left(\varphi_{12}\omega\right)\hbar'\left(\varphi_{12}\omega\right). \end{aligned} \tag{B.13}$$

Introducing notation

$$f_1(\omega) = P_{22}\hbar(\varphi_{22}\omega) - P_{11}\hbar(\varphi_{11}\omega) \tag{B.14}$$

$$f_2(\omega) = P_{11}\hbar(\varphi_{11}\omega) + P_{22}\hbar(\varphi_{22}\omega), \tag{B.15}$$

and calculating derivatives of interest

$$f_1'(\omega) = P_{22}\,\hbar'(\varphi_{22}\omega) - P_{11}\,\hbar'(\varphi_{11}\omega) \tag{B.16}$$

$$\mathrm{f}_1''(\omega) = P_{22}\hbar''(\varphi_{22}\omega) - P_{11}\hbar''(\varphi_{11}\omega) \tag{B.17}$$

$$f_2''(\omega) = P_{22}\hbar''(\varphi_{22}\omega) + P_{11}\hbar''(\varphi_{11}\omega), \tag{B.18}$$

the second derivative of the eigenvalue becomes

$$\begin{aligned}\frac{\partial^2}{\partial\,\omega^2}\sigma(\omega) = {}& \frac{1}{2}f_2''(\omega) \\ &+ \frac{1}{2}\frac{(f_1'(\omega))^2 + f_1(\omega)\,\mathrm{f}_1''(\omega)}{\sqrt{(f_1(\omega))^2 + 4\hbar(\varphi_{12}\omega)^2 P_{12}P_{21}}} \\ &+ 2P_{12}P_{21}\frac{(\hbar'(\varphi_{12}\omega))^2 + \hbar(\varphi_{12}\omega)\,\hbar''(\varphi_{12}\omega)}{\sqrt{(f_1(\omega))^2 + 4\hbar(\varphi_{12}\omega)^2 P_{12}P_{21}}} \\ &- \frac{1}{2}\frac{(f_1(\omega)\,f_1'(\omega) + 4P_{12}P_{21}\hbar(\varphi_{12}\omega)\,\hbar'(\varphi_{12}\omega))^2}{\left((f_1(\omega))^2 + 4\hbar(\varphi_{12}\omega)^2 P_{12}P_{21}\right)^{3/2}}.\end{aligned} \tag{B.19}$$

As the denominator of the expression (2.40) calls for evaluating (B.19) for $\omega = 0$, and as

$$f_1(0) = P_{22} - P_{11} \tag{B.20}$$

$$f_2(0) = P_{11} + P_{22} \tag{B.21}$$

$$f_1'(0) = 0 \tag{B.22}$$

$$\mathrm{f}_1''(0) = P_{22}\frac{\varphi_{22}^2}{3} - P_{11}\frac{\varphi_{11}^2}{3} \tag{B.23}$$

$$f_2''(0) = P_{22}\frac{\varphi_{22}^2}{3} + P_{11}\frac{\varphi_{11}^2}{3}, \tag{B.24}$$

we obtain

$$\left.\frac{\partial^2}{\partial\,\omega^2}\sigma\left(\omega\right)\right|_{\omega=0} = \frac{1}{3\left(P_{12}+P_{21}\right)}\left(P_{12}P_{22}\varphi_{22}^2 + P_{21}P_{11}\varphi_{11}^2 + 2P_{12}P_{21}\varphi_{12}^2\right). \qquad \text{(B.25)}$$

Defining

$$\begin{aligned} & w_1\left(\omega\right) = \\ & \left(P_{22}\hbar\left(\varphi^{22}\omega\right) - P_{11}\hbar\left(\varphi^{11}\omega\right)\right)\left(P_{22}\,\hbar'\left(\varphi^{22}\omega\right) - P_{11}\,\hbar'\left(\varphi^{11}\omega\right)\right) \\ & + 4\left(\hbar\left(\varphi^{12}\omega\right)\right)\left(\hbar'\left(\varphi^{12}\omega\right)\right)P_{12}P_{21}, \end{aligned} \qquad \text{(B.26)}$$

and

$$\mathrm{u}_1'\left(\omega\right) = f_1'\left(\omega\right) + \frac{w_1\left(\omega\right)}{\sqrt{\left(P_{22}\hbar\left(\varphi^{22}\omega\right) - P_{11}\hbar\left(\varphi^{11}\omega\right)\right)^2 + 4\left(\hbar\left(\varphi^{12}\omega\right)\right)^2 P_{12}P_{21}}}, \qquad \text{(B.27)}$$

the first derivatives of the eigenvector are

$$\frac{\partial}{\partial\,\omega}\nu_1\left(\omega\right) = 2P_{12}K\frac{\hbar'\left(\varphi^{12}\omega\right)u_1\left(\omega\right) - \left(\hbar\left(\varphi^{12}\omega\right)\right)\mathrm{u}_1'\left(\omega\right)}{\left(u_1\left(\omega\right)\right)^2} \qquad \text{(B.28)}$$

$$\frac{\partial}{\partial\,\omega}\nu_2\left(\omega\right) = 0. \qquad \text{(B.29)}$$

The first derivatives of the eigenvector evaluated for $\omega = 0$,

$$\lim_{\omega\to 0}\frac{\partial}{\partial\,\omega}\nu_1\left(\omega\right) = 0 \qquad \text{(B.30)}$$

$$\lim_{\omega\to 0}\frac{\partial}{\partial\,\omega}\nu_2\left(\omega\right) = 0 \qquad \text{(B.31)}$$

$$\lim_{\omega\to 0}\mathrm{u}_1'\left(\omega\right) = 0. \qquad \text{(B.32)}$$

Introducing

$$\begin{aligned}\mathrm{w}_1'(\omega) &= \left(P_{22}\,\hbar'\left(\varphi^{22}\omega\right) - P_{11}\,\hbar'\left(\varphi^{11}\omega\right)\right)^2 \\ &+ \left(P_{22}\hbar\left(\varphi^{22}\omega\right) - P_{11}\hbar\left(\varphi^{11}\omega\right)\right)\left(P_{22}\hbar''\left(\varphi^{22}\omega\right) - P_{11}\hbar''\left(\varphi^{11}\omega\right)\right) \\ &+ 4\left(\hbar'\left(\varphi^{12}\omega\right)\right)^2 P_{12}P_{21} + 4\left(\hbar\left(\varphi^{12}\omega\right)\right)\left(\hbar''\left(\varphi^{12}\omega\right)\right)P_{12}P_{21}, \qquad \text{(B.33)}\end{aligned}$$

$$s_q(\omega) = \sqrt{\left(P_{22}\hbar\left(\varphi^{22}\omega\right) - P_{11}\hbar\left(\varphi^{11}\omega\right)\right)^2 + 4\left(\hbar\left(\varphi^{12}\omega\right)\right)^2 P_{12}P_{21}},$$

and

$$\begin{aligned}u_1''(\omega) &= \frac{d^2}{d\,\omega^2}u_1(\omega) \\ &= \mathrm{f}_1''(\omega) + \frac{\mathrm{w}_1'(\omega)\,s_q(\omega) - w_1(\omega)\,\frac{w_1(\omega)}{s_q(\omega)}}{\left(s_q(\omega)\right)^2} \qquad \text{(B.34)}\end{aligned}$$

the second derivatives of the eigenvector elements are

$$\begin{aligned}\frac{d^2}{d\,\omega^2}\nu_1(\omega) &= 2KP_{12}\frac{\left(\hbar''\left(\varphi^{12}\omega\right)u_1(\omega) - \left(\hbar\left(\varphi^{12}\omega\right)\right)u_1''(\omega)\right)\left(u_1(\omega)\right)^2}{\left(u_1(\omega)\right)^4} \qquad \text{(B.35)} \\ &- 2KP_{12}\frac{\left(\hbar'\left(\varphi^{12}\omega\right)u_1(\omega) - \left(\hbar\left(\varphi^{12}\omega\right)\right)\mathrm{u}_1'(\omega)\right)2u_1(\omega)\,\mathrm{u}_1'(\omega)}{\left(u_1(\omega)\right)^4} \qquad \text{(B.36)} \\ \frac{d^2}{d\,\omega^2}\nu_2(\omega) &= 0, \qquad \text{(B.37)}\end{aligned}$$

and as (2.40) calls for the second derivatives evaluated for $\omega = 0$:

$$\lim_{\omega\to 0}\frac{d^2}{d\,\omega^2}\nu_2(\omega) = 0 \qquad \text{(B.38)}$$

$$\lim_{\omega\to 0}\frac{d^2}{d\,\omega^2}\nu_1(\omega) = \frac{K}{3\left(P_{12}+P_{21}\right)}\left(P_{11}\varphi_{11}^2 - P_{22}\varphi_{22}^2 + \left(P_{12}-P_{21}\right)\varphi_{12}^2\right). \qquad \text{(B.39)}$$

Inserting those values, as well as (2.38), in (2.40) we obtain

$$E\left[T_{\varphi_o}|t=1\right] = \frac{3\left(P_{12}+P_{21}\right)\varphi_o^2 + \left(\pi_1 - 1\right)\left(P_{11}\varphi_{11}^2 - P_{22}\varphi_{22}^2 + \left(P_{12}-P_{21}\right)\varphi_{12}^2\right)}{P_{12}P_{22}\varphi_{22}^2 + P_{21}P_{11}\varphi_{11}^2 + 2P_{12}P_{21}\varphi_{12}^2} \tag{B.40}$$

$$E\left[T_{\varphi_o}|t=2\right] = \frac{3\left(P_{12}+P_{21}\right)\varphi_o^2 + \pi_1\left(P_{11}\varphi_{11}^2 - P_{22}\varphi_{22}^2 + \left(P_{12}-P_{21}\right)\varphi_{12}^2\right)}{P_{12}P_{22}\varphi_{22}^2 + P_{21}P_{11}\varphi_{11}^2 + 2P_{12}P_{21}\varphi_{12}^2}, \tag{B.41}$$

where π_1 is the stationary probability of the state 1, and $E\left[T_{\varphi_o}|t=1\right]$, $E\left[T_{\varphi_o}|t=2\right]$ denote the expected number of hops before the spoke goes off course given the initial state of the Markov Chain is 1 and 2, respectively.

Replacing from (2.11) $P_{11}=0$ and $P_{12}=1$ we obtain the following

$$E\left[T_{\varphi_o}|t=1\right] = \frac{3(1+2P_{21}/P_{22})\varphi_o^2 - \frac{1+P_{21}/P_{22}}{1+2P_{21}/P_{22}}\left(\varphi_{12}^2-\varphi_{22}^2\right)}{\varphi_{22}^2+2P_{21}/P_{22}\varphi_{12}^2} \tag{B.42}$$

$$E\left[T_{\varphi_o}|t=2\right] = \frac{3(1+2P_{21}/P_{22})\varphi_o^2 + \frac{P_{21}/P_{22}}{1+2P_{21}/P_{22}}\left(\varphi_{12}^2-\varphi_{22}^2\right)}{\varphi_{22}^2+2P_{21}/P_{22}\varphi_{12}^2}, \tag{B.43}$$

Note that both (B.42) and (B.43) are functions of the terms P_{21}/P_{22} and φ_{i2}^2, where $i=1,2$. In the following we expand and approximate those terms, and illustrate that the expected number of hops until angular deviation hits the threshold depends on q only. We approximate the interior crescent angle with the following expression

$$\varphi_{i2}^2 = 2\frac{\Delta_i}{r}, \tag{B.44}$$

where $\Delta_i = r + h_i - R$. For two-state MC, applying (A.21), for $h_1 = R/2$ and $h_2 = r$, we obtain

$$\varphi_{i2}^2 = i\left(2-q\right) \tag{B.45}$$

$$\Delta_i/r = i\frac{2-q}{2}, \text{for } i=1,2. \tag{B.46}$$

For such small angles we can approximate the crescent areas $c_1 = c_{21}$ and $c_2 = c_{22}$ with expressions based on (A.24)

$$c_1 = \sqrt{2}\frac{R^2}{2}\left(\frac{(2-q)}{q}\right)^{3/2} = \frac{r^2}{f}(q)\frac{\sqrt{q}}{4},$$
$$c_2 = \sqrt{2}r^2\left(2-q\right)^{3/2} = r^2 f(q). \tag{B.47}$$

Applying (2.10) to expand the expressions for P_{21} and P_{22} defined in (2.11), we obtain that

$$\frac{P_{21}}{P_{22}} = \frac{c_1}{c_2 - c_1} = \frac{\frac{\sqrt{q}}{4}}{1-\frac{\sqrt{q}}{4}}. \tag{B.48}$$

Replacing (B.48) and (B.44) in (B.42) and (B.43), we obtain that the expected number of hops depends on q only.

Now we show that the terms related to the initial state of the Markov Chain in (2.40) can be ignored. For a two-state Markov Chain we consider,

$$\mu_1 = E\left[\mu_{i(T_{\varphi_o})}(\omega)\right]\Big|_{\omega=0} - \mu_1(\omega)|_{\omega=0}\,, \tag{B.49}$$

and

$$\mu_2 = E\left[\mu_{i(T_{\varphi_o})}(\omega)\right]\Big|_{\omega=0} - \mu_2(\omega)|_{\omega=0}\,, \tag{B.50}$$

for initial states 1 and 2, respectively. Starting from

$$E\left[\mu_{i(T_{\varphi_o})}(\omega)\right]\Big|_{\omega=0} = \pi_1\mu_1(0) + \left(1-\pi_1\right)\mu_2(0),$$

and as $\mu_2(0) = \lim_{\omega\to 0}\frac{d^2}{d\omega^2}\nu_2\left(\omega\right) = 0$, we obtain

$$\mu_1 = \left(\pi_1 - 1\right)\left.\frac{\nu_1''\left(\omega\right)}{\nu_1\left(\omega\right)}\right|_{\omega=0}, \tag{B.51}$$

and

$$\mu_2 = \pi_1\left.\frac{\nu_1''\left(\omega\right)}{\nu_1\left(\omega\right)}\right|_{\omega=0}. \tag{B.52}$$

Further, we replace the transition probabilities P_{11} and P_{12} in (B.39) with the values defined in (2.11), and obtain

$$\left.\frac{\nu_1''(\omega)}{\nu_1(\omega)}\right|_{\omega=0} = \frac{P_{22}}{3\left(1+P_{21}\right)}\left(\varphi_{12}^2-\varphi_{22}^2\right). \tag{B.53}$$

With stationary probability of state 1 defined as

$$\pi_1 = \frac{P_{21}}{1+P_{21}}, \tag{B.54}$$

we have

$$\mu_{\mathbf{1}} = \frac{P_{22}}{3\left(2-P_{22}\right)^2}\left(\varphi_{22}^2-\varphi_{12}^2\right) \tag{B.55}$$

$$\mu_{\mathbf{2}} = \frac{{P_{22}}^2-P_{22}}{3\left(2-P_{22}\right)^2}\left(\varphi_{22}^2-\varphi_{12}^2\right) = \left(P_{22}-1\right)\mu_{\mathbf{1}}. \tag{B.56}$$

We claim that, for large enough r, P_{22} is close to one. This makes $\mu_{\mathbf{2}}$ almost zero, if $\mu_{\mathbf{1}}$ is small as well. As the subtending angles φ_{i2} are both small, as shown in (B.45), and $\frac{P_{22}}{3(2-P_{22})^2} \approx 1/3$, the term $\mu_{\mathbf{1}}$ can be considered small ($(2-q)/3$), especially with respect to var $\left[\Theta_{T_{\varphi_o}}\right]$ in the denominator of (B.57).

$$E\left[T_{\varphi_o}\right] = \frac{\text{var}\left[\Theta_{T_{\varphi_o}}\right]+\mu_{\mathbf{1}}}{\frac{1}{3}\left(\pi_2P_{22}\varphi_{22}^2+\pi_1P_{11}\varphi_{11}^2+\pi_1P_{12}\varphi_{12}^2+\pi_2P_{21}\varphi_{21}^2\right)}. \tag{B.57}$$

B.1 MMRW Overshoot Analysis

We now approximate the Markov Process which modulates the MMRW discussed in 2.4 with a one-state Markov Chain. In this case, the modulated random walk $\Theta_n = \sum_{i=1}^{n}\Phi_i$ is in fact the IID random walk. This random walk stops if the condition in (6.18) is satisfied.

We define the undershoot as $X = \varphi_o - \Theta_{T_{\varphi_o}-1}$, while the overshoot is defined as $Y = \Theta_{T_{\varphi_o}} - \varphi_o$. As the IID Φ_i is uniform over $[-\varphi_{11}, \varphi_{11}]$, and as at T_{φ_o} Φ_i assumes a positive value, we conjecture that random variables X and Y have the same pdfs $f_X(x) = f_Y(x)$ (or at least the first two moments), both uniform, with support set $[0, \varphi_{11}]$. We define random variable

$Z = X + Y$ s.t. $Z|Y \sim U\left(Y, \varphi_{11}\right)$.

As

$$E\left[Z\right] = E\left[Y\right] + E\left[X\right] = 2E\left[Y\right] = 2m$$

and

$$E\left[Z\right] = E_Y\left\{E\left[Z|Y\right]\right\} = E_Y\left\{(Y + \varphi_{11})/2\right\} = (m + \varphi_{11})/2,$$

we obtain the first moment of the overshoot as $E\left[Y\right] = m = \varphi_{11}/3$. Further, we establish

$$\begin{aligned} E\left[Y^2\right] &= m_2, E\left[Z^2\right] = 2m_2 + 2E\left[XY\right] = m_2 + m\varphi_{11} \\ E\left[Z^2\right] &= E_Y E\left[Z^2|Y\right] = (1/3)(m_2 + m\varphi_{11} + \varphi_{11}^2) \end{aligned} \tag{B.58}$$

Solving the system of equations (B.58) we obtain the second moment of the overshoot $E\left[Y^2\right] = \varphi_{11}^2/6$. For symmetry reasons the variance of the random walk at overshoot is equal at both φ_o and $-\varphi_o$. Thus, as both overshoot occurrences are equiprobable,

$$\begin{aligned} \text{var}\left[\Theta_{T_{\varphi_o}}\right] &= 0.5\left(2E\left[(\varphi_o + Y)^2\right]\right) \\ &= \varphi_o^2 + (2/3)\varphi_o\varphi_{11} + \varphi_{11}^2/6. \end{aligned} \tag{B.59}$$

Appendix C
Adaptive Mechanism Derivations

When calculating the average probability of a hop in One-Step Backward Repair Model, we approximate the crescent areas with the appropriate rectangular areas. The area of the envelope over which the average is taken, is calculated as the length of its lower boundary times its width, which equals $(r - R/2)$ for $m = 1$, and $(2r - R)$ for $m = 2$. The same stands for the "'inserted"' (failed) crescent: the area is the product of its lower boundary and the width. We distinguish between two distinct areas, $s_S \approx l_{cs}\,(r - R/2)$, and $s_L \approx l_{cl}\,(2r - R)$, approximating the small and the large crescent area, respectively. Here, l_{cs} and l_{cl} are the lower boundaries of the respective crescents.

From Figure C.1, for $p = 1$, $m = 2$, $n = 2$, we observe that the envelope length is

$$l_E = R(\cos^{-1}\left(R^2 - 2r^2\right)/\left(2r^2\right) + 2\cos^{-1}\left(3R^2 - 4r^2\right)/\left(4rR\right))$$

and the crescent length is

$$l_{cl} = R\cos^{-1}\left(R^2 - 2r^2\right)/\left(2r^2\right)$$

.

In order to provide a universal representation of all the different cases of envelopes and relative positions of the failed relay, we now introduce the following notation

$$l_E = l_c\,(2 + \xi)\,, \tag{C.1}$$

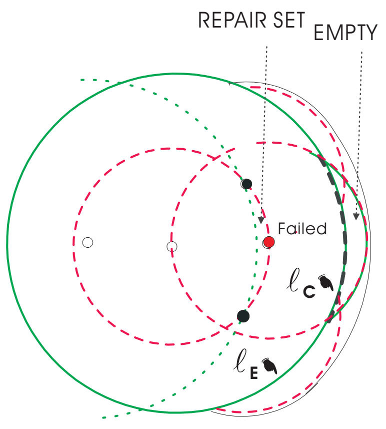


Figure C.1: Adaptive Envelopes for Pivot State 1: Large envelope, large failed crescent (replacement must stay in state 2, failed CLR in state 2). The pointers indicate the length of the envelope ℓ_E and the length of the failed crescent ℓ_c.

where, for $p = 1$, $m = 2$ and $n = 2$,

$$\xi = 2\left(\cos^{-1}\left(3R^2 - 4r^2\right)/\left(4rR\right)\right)/\left(\cos^{-1}\left(R^2 - 2r^2\right)/\left(2r^2\right)\right) - 1$$

and $l_c = l_{cl}$.

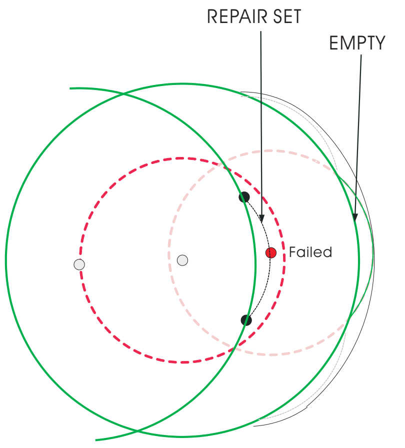
Note the pointers in Figure C.1 showing the envelope length l_E as the length of the large circular segment at the bottom of the envelope, and l_c as the length of the smaller circular segment. For other envelope cases, l_E and l_c refer to the analogous circular segments.

Next, in Figure C.2 (a), for $p = 2$, $m = 1$, $n = 1$, the envelope length is

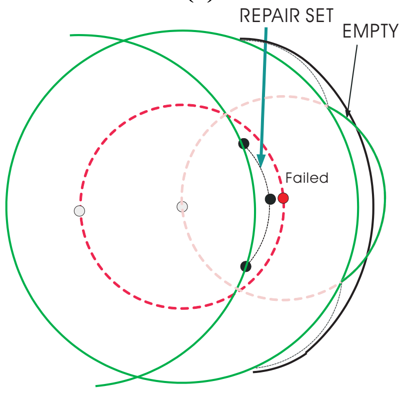
$$l_E = R(2\cos^{-1}\left(5R^2 - 4r^2\right)/\left(4R^2\right) + 2\cos^{-1}\left(3R^2 - 4r^2\right)/\left(4rR\right))$$

and the crescent length is

$$l_{cs} = 2R\cos^{-1}\left(5R^2 - 4r^2\right)/\left(4R^2\right)$$



(a)



(b)

Figure C.2: Small Adaptive Envelopes for Pivot State 2 (Replacement Relay in State 1): (a)Small envelope, small crescent (failed CLR also in state 1). (b) Small envelope, large crescent (failed CLR in state 2).

; hence,

$$l_E = l_c \left(2 + \xi\right), \tag{C.2}$$

where now

$$\xi \quad = \quad \frac{\cos^{-1}\frac{3R^2-4r^2}{4rR} - \cos^{-1}\frac{5R^2-4r^2}{4R^2}}{\cos^{-1}\frac{5R^2-4r^2}{4R^2}} \tag{C.3}$$

and $l_c = l_{cs}$.

In Figure C.3 (a), for $p = 2$, $m = 2$, $n = 2$, the envelope length is

$$l_E = 3R\cos^{-1}\left(R^2 - 2r^2\right) / \left(2r^2\right) = 3l_c$$

and the crescent length is

$$l_{cl} = R\cos^{-1}\left(R^2 - 2r^2\right) / \left(2r^2\right)$$

. Hence, we have

$$l_E = l_c \left(2 + \xi\right), \qquad \xi = 1 \text{ and } l_c = l_{cl}. \tag{C.4}$$

Figure C.2 (b) and Figure C.3 (b) respectively illustrate cases when the failed relay was from state $n = 2$, and the relay attempting to repair is from state $m = 1$, and vice versa. The probability of the repair success, for the case presented in Figure C.2 (b), can be approximated with the probability of success when both relays are from state $m = n = 1$ (Figure C.2 (a)).

For the case presented in Figure C.3 (b), i.e. $n = 1$, $m = 2$, note that the probability of repair success is larger than the probability of success when both relays (i.e. m and n) are in state 2 (Figure C.3 (a)), as the innovation area is larger by a constant factor $\tilde{s} = s_L - s_S$.

Having introduced the universal notation, we calculate the average probability of repair success according to the following formula

$$\begin{aligned}
\overline{e^{in}_{p,m,n}} &= E\left[(1-e^{-S^{m,n}_{in}(k+1)})\right]_{\Xi(pm)} \\
&= \frac{1}{(1+\xi)^2 {l_c}^2} \\
&\quad \left(\int_0^{l_c}\int_0^{y}\left(1-e^{-\left(\frac{y-x}{l_c}s_1+\tilde{s}\right)}\right)dx\,dy\right. \\
&\quad +\int_{l_c}^{(1+\xi)l_c}\int_{y-l_c}^{y}\left(1-e^{-\left(\frac{y-x}{l_c}s_1+\tilde{s}\right)}\right)dx\,dy \\
&\quad \left.+\int_{l_c}^{(1+\xi)l_c}\int_{0}^{y-l_c}\left(1-e^{-s_2}\right)dx\,dy\right) \\
&\quad +\frac{1}{(1+\xi)^2 {l_c}^2} \\
&\quad \left(\int_0^{\xi l_c}\int_y^{y+l_c}\left(1-e^{-\left(\frac{x-y}{l_c}s_1+\tilde{s}\right)}\right)dx\,dy\right. \\
&\quad \left.+\int_{\xi l_c}^{(1+\xi)l_c}\int_{y}^{(1+\xi)l_c}\left(1-e^{-\left(\frac{x-y}{l_c}s_1+\tilde{s}\right)}\right)dx\,dy\right) \\
&\quad +\frac{1}{(1+\xi)^2 {l_c}^2} \\
&\quad \left(\int_0^{\xi l_c}\int_{y+l_c}^{(1+\xi)l_c}\left(1-e^{-s_2}\right)dx\,dy\right),
\end{aligned} \tag{C.5}$$

where

$$\begin{aligned}
&\xi = 2\tfrac{\varphi^{12}}{\varphi^{22}} - 1 \text{ for } p=1, m=2 \\
&\xi = \frac{\varphi^{12}-\cos^{-1}\left(\frac{5R^2-4r^2}{4R^2}\right)}{\cos^{-1}\left(\frac{5R^2-4r^2}{4R^2}\right)} \text{ for } p=2, m=1 \\
&\xi = 1 \text{ for } p=2, m=2 \\
&s_1 = s_L \text{ and } s_2 = s_L \text{ for } p=1, m=2 \\
&s_1 = s_S \text{ and } s_2 = s_S \text{ for } p=2, m=1 \\
&s_1 = s_L \text{ and } s_2 = s_L \text{ for } p=2, m=2, n=2 \\
&s_1 = s_S \text{ and } s_2 = s_L \text{ for } p=2, m=2, n=1.
\end{aligned}$$

Here, $\varphi^{ij} = \beta(h_i, h_j)$, denotes the angular displacement associated with transition from state i to state j (when the previous hop length is h_i, and the current length is h_j); Also, $\tilde{s} = s_2 - s_1$.

Solving the integrals in (C.5) results in a closed-form expression for the probability of repair

$$\begin{aligned}\overline{e^{in}_{p,m,n}} = &\frac{\xi^2}{(1+\xi)^2}\left(1-e^{-s_2}\right)+\\ &\frac{2}{(1+\xi)^2}\left(0.5-\frac{e^{-\tilde{s}}}{s_1}+\frac{e^{-\tilde{s}}}{s_1^2}\left(1-e^{-s_1}\right)\right.\\ &\left.+\xi-\xi\frac{e^{-\tilde{s}}}{s_1}(1-e^{-s_1})\right).\end{aligned} \tag{C.6}$$

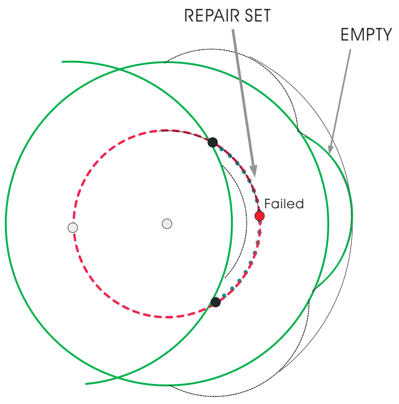
The following algorithm calculates the transition probabilities of the Markov Chain that models the BeSpoken enhanced with One-Step Backward Repair mechanism.

```
$P^A_{pm} = P^B_{pm}$
for $n = (1,2)$
    {
    $P^A_{pm} += S_T(p,n)e^{-S_c(2,n)}(1-e^{-(S_c(2,p)-1)})S_T(p,m)\overline{e^{in}_{p,m,n}}$
    }                                                                    (C.7)
$P^A_{11} = 0,$                                                          (C.8)
```

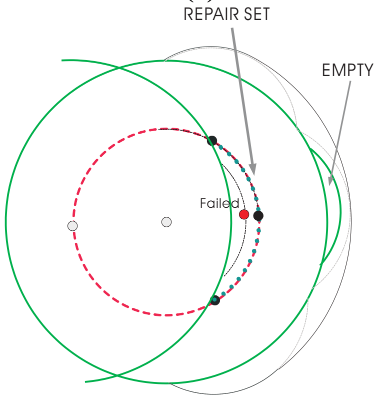
where $P^B_{pm} = \breve{P}_{pm}$ are the following transition probabilities, related to the first attempt to find the next relay, i.e. without utilizing the adaptive mechanism:

$$\begin{aligned}P^B_{11} &= 0\\ P^B_{12} &= \overline{e_L}\\ P^B_{21} &= \frac{s_1}{s_L}\overline{e_S}\\ P^B_{22} &= \frac{s_2}{s_L}\overline{e_L},\end{aligned} \tag{C.9}$$

and coefficients $S_T(p,n)$ and $S_c(p,n)$ are the elements of two matrices associated with the two-state adaptive Markov Chain where the two-level uniform quantization approximation is



(a)



(b)

Figure C.3: Large Adaptive Envelopes for Pivot State 2 (Replacement Relay in State 2): (a) Large envelope, large failed crescent (failed CLR in state 2). (b) Large envelope, small failed crescent (failed CLR in state 1).

applied, in that $s_1 = s_S$, and $s_2 = s_L - s_S$

$$\mathbf{S_C} = \begin{bmatrix} 0 & s_L - s_S \\ s_S & s_L - s_S \end{bmatrix}, \tag{C.10}$$

$$\mathbf{S_T} = \begin{bmatrix} 1 & 1 \\ s_S/s_L & 1 - s_S/s_L \end{bmatrix}. \tag{C.11}$$

Appendix D

Random Walk Ripple Evolution: The Stopping Time Probability

Recall that for the Markov Chain model of the ripple evolution, described by (6.19) and (6.20), its trapping state corresponds to the empty ripple. The probability of entering the trapping state at time t, where $t > T_i$, is given in (6.22), where $u = t - T_i$. The probability of being in the trapping state at $T_i{+}u$ can also be expressed as $p^{(T_i)}_{T_i+u} = [0\;0\;1\;0\cdots 0]\,\mathbf{P}_i^{(u-1)}\,[1\;\eta(0)\;0\cdots 0]^T$. Hence, we can reformulate (6.22) as

$$p^{T_i}(u) \quad = \quad [0\;0\;1\;0\cdots 0]\,\mathbf{P}_i^{u-1}\,[0\;\eta(0)\;0\cdots 0]^T\,. \tag{D.1}$$

Note that both $[0\;0\;1\;0\cdots 0]$ and $[0\;\eta(0)\;0\cdots 0]^T$ have zero-valued first elements, which means that the first row and the first column of the transition probability matrix $\mathbf{P}_i$ do not contribute to the value of (D.1). Hence, we introduce a new matrix $\widetilde{\mathbf{P}}_i$ which contains the significant elements of $\mathbf{P}_i$ as

$$\widetilde{\mathbf{P}}_\ell \quad = \quad \begin{bmatrix} \eta(1) & \eta(2) & \eta(3) & \cdots & 0 \\ \eta(0) & \eta(1) & \eta(2) & \cdots & 0 \\ 0 & \eta(0) & \eta(1) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \eta(0) & \eta(1) \end{bmatrix}_{k-1\times k-1} \quad , \tag{D.2}$$

with $\eta\,(\cdot) \equiv \wp\left(\lambda_\ell^{(\delta)}\right)$. Now,

$$p^{T_i}(u) \quad = \quad \eta(0)\,[0\;1\;0\cdots 0]\,\widetilde{\mathbf{P}}_i^{(u-1)}\,[1\;0\;0\cdots 0]^T\,. \tag{D.3}$$

Assuming n is large, we can approximately express the uth power of the matrix $\widetilde{\mathbf{P}}_i$ through a matrix that contains elements $\aleph^{(u)}()$ of the uth convolution of the pdf array $\overline{\eta} = [\eta(0)\;\eta(1)\;\cdots]$.

Let us define $\overline{\eta}$ as degree-one convolution. For order-two convolution, we convolve $\overline{\eta}$ with itself, and uth convolution of $\overline{\eta}$ is obtained by recursively convolving $(u-1)$th convolution with $\overline{\eta}$. By multiplying the matrix

$$\widetilde{\mathbf{P}}_i^C = \begin{bmatrix} \eta(0) & \eta(1) & \eta(2) & \cdots \\ 0 & \eta(0) & \eta(1) & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \eta(0) & \eta(1) \end{bmatrix}, \tag{D.4}$$

which was obtained by adding the column $[\eta(0)\ 0\ 0\ \cdots]^T$ in front of $\widetilde{\mathbf{P}}_i$, and another matrix

$$\widetilde{\mathbf{P}}_i^R = \begin{bmatrix} \eta(2) & \eta(3) & \eta(4) & \cdots \\ \eta(1) & \eta(2) & \eta(3) & \cdots \\ \eta(0) & \eta(1) & \eta(2) & \cdots \\ 0 & \eta(0) & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}, \tag{D.5}$$

which was obtained by adding the row $[\eta(2)\ \eta(3)\ \eta(4)\cdots]$ above $\widetilde{\mathbf{P}}_i$, we obtain

$$\widetilde{\mathbf{P}}_i^C \widetilde{\mathbf{P}}_i^R = \begin{bmatrix} \aleph^{(2)}(2) & \aleph^{(2)}(3) & \cdots & \\ \aleph^{(2)}(1) & \aleph^{(2)}(2) & \cdots & \\ \aleph^{(2)}(0) & \aleph^{(2)}(2) & \cdots & \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix} \tag{D.6}$$

$$= \widetilde{\mathbf{D}}^{(2)}, \tag{D.7}$$

where $\aleph^{(s)}(d)$ is the s-th convolution of $\eta(\cdot)$ evaluated at d, and $\widetilde{\mathbf{D}}^{(2)}$ is what we refer to as second convolution matrix of $\overline{\eta}$, for $\eta(\cdot) \equiv \wp\left(\lambda_{t_i}^{(\delta)}\right)$. Hence,

$$\widetilde{\mathbf{P}}_i^2 = \widetilde{\mathbf{D}}^{(2)} - \eta(0) \begin{bmatrix} \eta(2) & \eta(3) & \cdots \\ 0 & 0 & \cdots \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{D.8}$$

$$= \widetilde{\mathbf{D}}^{(2)} - [\eta(0)\ 0\ 0\cdots]^T \left[\aleph^{(1)}(2)\ \aleph^{(1)}(3)\ \aleph^{(1)}(4)\cdots\right]. \tag{D.9}$$

By induction,

$$\begin{aligned}
\widetilde{\mathbf{P}}_i^3 &= \widetilde{\mathbf{D}}^{(3)} - [\eta(0)\;0\;0\cdots]^T \left[\aleph^{(2)}(3)\;\aleph^{(2)}(4)\;\aleph^{(2)}(5)\cdots\right] \\
&- \widetilde{\mathbf{P}}_i\,[\eta(0)\;0\cdots]^T \left[\aleph^{(1)}(2)\;\aleph^{(1)}(3)\;\aleph^{(1)}(4)\cdots\right], \\
\widetilde{\mathbf{P}}_i^u &= \widetilde{\mathbf{D}}^{(u)} - \sum_{z=2}^{u} \widetilde{\mathbf{P}}_i^{(u-z)}\,[\eta(0)\;0\cdots]^T \left[\aleph^{(z-1)}(z)\;\aleph^{(z-1)}(z+1)\;\aleph^{(z-1)}(z+2)\cdots\right].
\end{aligned} \tag{D.10}$$

Replacing (D.8) in (D.3), we obtain (6.23).

References

[1] Akyildiz I.F., Su W., Sankarasubramaniam Y., and Cayirci E. A survey on sensor networks. *IEEE Communications Magazine*, 40:102–116, 2002.

[2] N. Bulusu, J. Heidemann, and D. Estrin. Gps-less low cost outdoor localization for very small devices. *Personal Communications, Special Issue on "Smart Spaces and Environments"*, 7, Oct 2000.

[3] Darren Griffin. How does the global positioning system work ? http://www.pocketgpsworld.com/howgpsworks.php, 2008.

[4] J. C. Navas and T. Imielinski. Geocast - geographic addressing and routing. In *ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom)*, Sep 1997.

[5] Andreas Willig Holger Karl. *Protocols and Architectures for Wireless Sensor Networks*. J. Wiley, 2005.

[6] J. Wieselthier, G. Nguyen, and A. Ephremides. On the construction of energy-efficient broadcast and multicast trees in wireless networks. In *IEEE INFOCOM00*, Mar 2000.

[7] I. Maric and R. D. Yates. Cooperative multihop broadcast for wireless networks. *IEEE JSAC Special Issue on Fundamental Performance Limits of WSNs*, 22, Aug 2004.

[8] S. Katti, Dina Katabi, W. Hu, H. Rahul, and M. Medard. The importance of being opportunistic: Practical network coding for wireless environments. In *43nd Annual Allerton Conference*, Sep 2005.

[9] G. Barrenechea, B. Beferull-Lozano, and M. Vetterli. Lattice sensor networks: capacity limits, optimal routing and robustness to failures. In *IPSN 2004*, Apr 2004.

[10] L. Sankaranarayanan, G. Kramer, and N. Mandayam. Hierarchical sensor networks: Capacity bounds and cooperative strategies using the multiple-access relays. In *IEEE SECON 2004*, October 2004.

[11] R. C. Shah and J. Rabaey. Energy aware routing for low energy ad hoc sensor networks. In *IEEE Wireless Communications and Networking Conference 2002*, 2002.

[12] Y. Yu, D. Estrin, and R. Govindan. Geographical and energy-aware routing: A recursive data dissemination protocol for wireless sensor networks. Technical report, UCLA Computer Science Department, 2001.

[13] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *Proceedings of 7th Annual International Conference on Mobile Computing and Networking*, pages 85–96, July 2001.

[14] A. Cerpa and D. Estrin. Ascent: Adaptive self-configuring sensor network topologies. In *Proceedings of INFOCOM 2002*, June 2002.

[15] S. Funke and C. Klein. Hole detection or: how much geometry hides in connectivity? In *The 22nd annual symposium on Computational geometry*, 2006.

[16] Q. Fang, J. Gao, L. Guibas, V. de Silva, and L. Zhang. Glider:gradient landmark-based distributed routing for sensor networks. In *Proc. of the 24th Conference of the IEEE Communication Society (INFOCOM)*, Mar 2005.

[17] R. Fonesca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, and I. Stoica. Beacon vector routing: Scalable point-to-point routing in wireless sensornets. In *Proc. of the 2nd Symposium on Networked Systems Design and Implementation*, 2005.

[18] A. Caruso, A. Urpi, S. Chessa, and S. De. Gps free coordinate assignment and routing in wireless sensor networks. In *Proc. of the 24th INFOCOM*, Mar 2005.

[19] S. Ni, Y. Tseng, Y. Chen, and J. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proc. 5th ACM/IEEE MOBICOM*, 1999.

[20] F. Bai and A. Helmy. Comparative analysis of algorithms for tree structure restoration in sensor networks. In *Proc. of 23rd IEEE ICPCC*, pages 385–391, 2004.

[21] John Heidemann, Fabio Silva, and Deborah Estrin. Matching data dissemination algorithms to application requirements. In *ACM SenSys Conference, Los Angeles, California, USA*, pages 218–229, Nov 2003.

[22] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking (TON)*, 11(1):2–16, 2003.

[23] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Proc. 1st ACM Int'l. Wksp. Wireless Sensor Nets and Apps.*, 2002.

[24] X. Liu, Qingfeng Huang, and Ying Zhang. Combs, needles, haystacks: Balancing push and pull for discovery in large-scale sensor networks. In *ACM Sensys 2004*, November 2004.

[25] B. Karp and H. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proc. of the 6th MOBICOM*, 2000.

[26] H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Trans. on Comms.*, 1984.

[27] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *Proc. of IEEE MobiCom 2003*, 2003.

[28] B. Leong, B. Liskov, and R. Morris. Greedy virtual coordinates for geographic routing. In *Proc. of ICNP 2007*, 2007.

[29] J. Li, J. Jannotti, D. DeCouto, D. Karger, and R. Morris. A scalable location service for geographic ad-hoc routing. In *Proc. of the 6th MOBICOM*, 2000.

[30] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. Ght: A geographic hash table for data-centric storage in sensornets. In *1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.

[31] P. Bose, P. Morin, I. Stojmenovic, , and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Proc. of the 3rd Int. Workshop on Discrete Algorithms and methods for mobile computing and communications (DialM)*, 1999.

[32] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86, 1990.

[33] H. Breu and D. G. Kirkpatrick. Unit disk graph recognition is np-hard. *Computational Geometry Theory and Applications*, 9, 1998.

[34] A. Cerpa, J. L. Wong, L. Kuang, M. Potkonjak, and D. Estrin. Statistical model of lossy links in wireless sensor networks. In *Proceedings of the 4th IPSN*, 2005.

[35] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein. Growth codes: Maximizing sensor network data persistence. In *ACM SIGCOMM 2006*, 2006.

[36] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran. Decentralized erasure codes for distributed networked storage. *IEEE/ACM Transactions on Networking*, 14, 2006.

[37] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran. Distributed fountain codes for networked storage. In *ICASSP 2006*, 2006.

[38] Y. Liu, B. Liang, and B. Li. Data persistence in large-scale sensor networks with decentralized fountain codes. In *IEEE Infocom*, May 2007.

[39] S. Aly, Z. Kong, and E. Soljanin. Fountain codes based distributed storage algorithms for large-scale wireless sensor networks. In *IPSN 2008*. IEEE, April 2008.

[40] S. Aly, Z. Kong, and E. Soljanin. Raptor codes based distributed storage algorithms for wireless sensor networks. In *ISIT 2008*. IEEE, July 2008.

[41] R. Nelson and L. Kleinrock. The spatial capacity of a slotted aloha multihop packet radio network with capture. *IEEE Trans. on Comms.*, 1984.

[42] T.C. Hou and V.O.K. Li. Transmission range control in multihop packet radio networks. *IEEE Trans. on Comms.*, 1986.

[43] A. Busson, G. Chelius, and E. Fleury. Energy aware unicast geographic routing. In *SpaSWIN 2006 workshop*. IEEE, April 2006.

[44] M. Mauve, J.Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad-hoc networks. *IEEE Network*, Nov 2001.

[45] B. Hajek. Minimum mean hitting times of brownian motion with constrained drift. In *Proc. 27th Conf. on Stochastic Processes and Their Applications*, 2000.

[46] S. Shakkottai. Asymptotics of query strategies over a sensor network. In *Proceedings 23rd IEEE Infocom, Hong Kong*, 2004.

[47] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk. Enhancing source-location privacy in sensor network routing. In *Proc. of the 25th Annual IEEE ICDCS*, pages 599–608, 2005.

[48] D. Niculescu and B. Nath. Trajectory based forwarding and its applications. In *Proc. 9th ACM/IEEE MOBICOM*, pages 260–272, 2003.

[49] D. Slepian and J. K. Wolf. Noiseless coding of correlated information sources. *IEEE Trans. on Information Theory*, 19, 1973.

[50] R.Gallager. *Discrete Stochastic Processes*. Kluwer Academic Publishers, 1995.

[51] A. Dembo and O. Zeitouni. *Large Deviation Techniques and Applications*. Springer, 1998.

[52] N. Ahmed, S. S. Kanhere, and S. Jha. The holes problem in wireless sensor networks: a survey. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9:418, 2005.

[53] S. Kokalj-Filipovic, P. Spasojevic, and R. Yates. Bespoken protocol for data dissemination in wireless sensor networks. In *SpaSWIN 2007 workshop*. IEEE, April 2007.

[54] A. Dimakis and K. Ramchandran. *Networked Sensing Information and Control*, chapter Network Coding for Distributed Storage in Wireless Networks. Springer US, 2008.

[55] S. Kokalj-Filipovic, P. Spasojevic, R. Yates, and E. Soljanin. Decentralized fountain codes for minimum-delay data collection. In *CISS 2008*. IEEE, March 2008.

[56] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros. The benefits of coding over routing in a randomized setting. In *IEEE International Symposium on Info. Theory*, Jun 2003.

[57] T. Ho, M. Medard, J. Shi, M. Effros, and D. R. Karger. On randomized network coding. In *41st Annual Allerton Conference*, Oct 2003.

[58] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.

[59] P. Elias, A. Feinstein, and C.E. Shannon. A note on the maximum flow through a network. *IRE Transactions on Information Theory*, pages 117–119, 1956.

[60] L.R. Ford, Jr., and D.R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics 8*, pages 399–404, 1956.

[61] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Trans. on Information Theory*, 46, 2000.

[62] T. Ho, M. Medard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong. A random linear network coding approach to multicast. *IEEE Transactions on Information Theory*, Oct 2006.

[63] W. Feller. *An Introduction to Probability Theory and Its Applications vol. 1*. John Willey and Sons, 1968.

[64] L. Lovasz. Random walks on graphs: a survey. *Combinatorics*, 2, 1993.

[65] W.K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrica*, 57, 1970.

[66] M. Luby. L t codes. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002.

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

[68] Z. Li and B. Li. Improving throughput in multihop wireless networks. *IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY*, 55, May 2006.

[69] D. H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory*, 1986.

[70] S. Sanghavi. Intermediate performance of rateless codes. In *Information Theory Workshop, 2007. ITW '07*, 2007.

[71] Venkatesan Guruswami. Rapidly mixing markov chains: A comparison of techniques, 2000.

[72] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Mixing times for random walks on geometric random graphs. In *SIAM ANALCO 2005*, 2005.

[73] Y.Wu, P.A.Chou, and S.-Y.Kung. Information exchange in wireless networks with network coding and physical-layer broadcast. In *CISS 2005*, 2005.

[74] J. Widmer, C. Fragouli, and J.-Y. Le Boudec. Low-complexity energy-efficient broadcasting in wireless ad-hoc networks using network coding. In *1st Network Coding Workshop (NetCod)*, April 2005.

[75] B. Hajek. Connections between network coding and stochastic network theory. In *Stochastic Networks Conference*, 2006.

[76] S. ten Brink. Code doping for triggering iterative decoding convergence. In *International Symposium on Information Theory*, 2001.

[77] G. Caire, S. Shamai, and S. Verdu. Feedback and belief propagation. In *4th Int. Symp. on Turbo Codes and Related Topics*, 2006.

[78] R. Karp, M. Luby, and A.Shokrollahi. Finite length analysis of lt codes. In *ISIT 2004*, 2004.

[79] E.Maneva and A.Shokrollahi. New model for rigorous analysis of lt-codes. In *ISIT 2006*, 2006.

Curriculum Vitae

Silvija Kokalj-Filipović

Education:

PhD Electrical and Computer Engineering

January 2009

MS Electrical and Computer Engineering

October 1995

Dipl. Ing. Electrical Engineering

June 1989

Work History:

January 2004 till present

Graduate Assistant at Winlab, Rutgers University

September 2002 till December 2003

Part-time Research Engineer at Winlab, Rutgers University

Summer 2002

Contract job with Digital5 , West Windsor, NJ

from March 2001 till March 2002

Senior Software Engineer at Wiscom Technologhies, Clark NJ

from April 2000 till February 2001

Senior Software Engineer at Eulix Networks, Princeton NJ

April 1999-April 2000

Senior Software Engineer at Ariel Corporation, Cranbury, NJ

January 1997 - October 1998

Senior Network Software Engineer/System Support Engineer - Geotek, Montvale, NJ

January 1996 - October 1996

Technical Staff Member at Formation, Inc., Moorestown, NJ

1989-1995

Research and Project Engineer at Informatika d.d., Belgrade, Yugoslavia

Journal Papers:

1. S. Kokalj-Filipović, P. Spasojević, and E. Soljanin. Doped fountain coding for minimum delay data collection in circular networks. *Submitted to IEEE JSAC - Special Issue on Network Coding for Wireless Communication Networks*, 2008.

2. S. Kokalj-Filipović, P. Spasojević, and R. Yates. Random walk models in protocol design for geographic data propagation in location-unaware wireless sensor networks. *Submitted to IEEE JSAC - Special Issue on Stochastic Geometry and Random Graphs for Wireless Networks*, 2008.

Conference Publications:

1. S. Kokalj-Filipović, P. Spasojević, E. Soljanin, and R. Yates. ARQ with doped fountain decoding. In *International Symposium on Spread Spectrum Techniques and Applications (ISSSTA) 2008*. IEEE, Bologna, August 2008.

2. S. Kokalj-Filipović, P. Spasojević, R. Yates, and E. Soljanin. Decentralized fountain codes for minimum-delay data collection. In *Conference on Information Sciences and Systems (CISS) 2008*. IEEE, Princeton, March 2008.

3. S. Kokalj-Filipović, P. Spasojević, and R. Yates. Bespoken protocol for data dissemination in wireless sensor networks. In *Third Workshop on Spatial Stochastic Models for Wireless Networks (SpaSWIN) 2007*. IEEE, Limassol, Cyprus, April 2007.

4. S. Kokalj-Filipović, R. Yates, and P. Spasojević. Random walk models for geographic data propagation in wireless sensor networks. In *Conference on Information Sciences and Systems (CISS) 2007*. IEEE, Baltimore, March 2007.

Conference Submissions:

1. S. Kokalj-Filipović, P. Spasojević, and R. Yates. Adaptive BeSpoken: Can a Packet Walk Straight Through a Field of Randomly Dying Location-Unaware Wireless Nodes? Submitted to *IPSN 2009*. ACM/IEEE , April 2009.