Coding and Message-Passing for Large-Scale Storage and Inference

Georgios Alexandros Dimakis

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2008-153
http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-153.html

December 10, 2008

Coding and Message-Passing for Large-Scale Storage and Inference

by

Georgios Alexandros Dimakis

Ptychion (National Technical University of Athens) 2003
M.S. (University of California, Berkeley) 2005

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

Doctor of Philosophy

in

Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Kannan Ramchandran, Chair
Professor Martin Wainwright
Professor Sourav Chatterjee

Fall 2008

The dissertation of Georgios Alexandros Dimakis is approved:

Professor Kannan Ramchandran, Chair	Date
Professor Martin Wainwright	Date
Professor Sourav Chatterjee	Date

University of California, Berkeley

Fall 2008

Coding and Message-Passing for Large-Scale Storage and Inference

by

Georgios Alexandros Dimakis

Abstract

Coding and Message-Passing for Large-Scale Storage and Inference

by

Georgios Alexandros Dimakis

Doctor of Philosophy in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Kannan Ramchandran, Chair

Recent advances in technology have catalyzed a paradigm shift away from centralized schemes and in the direction of distributed and cooperative architectures for large-scale systems. In applications like data centers, sensor networks, and peer-to-peer networks, coding is used to introduce redundancy for robustness. We develop and analyze novel coding constructions for distributed storage applications. We also present information theoretic performance bounds and explicit network codes that achieve optimal performance.

For the case of large-scale distributed processing, we introduce new message-passing schemes and show explicit results on convergence rate. In particular, we introduce geographic gossip, an algorithm that can compute linear functions of data in a network, requiring a number of messages that scales optimally in the number of nodes for a large class of graphs.

Professor Kannan Ramchandran, Chair Date

Acknowledgements

I feel that I have been tremendously lucky in my graduate life, because I had the opportunity to interact with numerous mentors and friends. Without them, this work would undoubtedly have not been possible. First and foremost I would like to thank my research advisors: Kannan Ramchandran and Martin Wainwright. I am deeply indebted to Kannan for first taking me as a Master's student, guiding me through my first steps in research and sharing his numerous ideas, many of which form the foundations of this thesis. I started working with Martin after he joined Berkeley in 2005. I want to thank him for sharing his incredible mathematical skill and for the countless hours we have spent together trying to crack down problems. Kannan and Martin have been essential for me to realize my potential and in addition have been mentors and good friends. I can only wish that I will be to my students as great an advisor as Kannan and Martin have been to me.

I would like to thank my collaborators: Chapter 2 is based on joint work with Vinod Prabhkaran and Chapter 4 with Anand Sarwate both of whom have been mentors and valuable friends. I spent the summer of 2005 in EPFL working with Florence Benezit, Martin Vetterli, and Patrick Thiran. Chapter 5 is based on our joint work and I am indebted to Florence, Patrick, and Martin for the hours we spent working together and the numerous things I learned from them.

I would like to thank Microsoft Research for the financial support through their research fellowship for the years 2007-2008 and also for the summer internship. Chapter 3 is based on joint work with Yunnan Wu from MSR and Brighten Godfrey from Berkeley. Yunnan and Brighten have been valuable collaborators and my experience at MSR has given me a valuable connection to real world problems that will shape my future research. I would like to thank Phil Chou and everybody in the Com-

munication and Collaboration group for numerous discussions and a very creative summer.

I would also like to thank my collaborators on projects that did not appear in this thesis: Costis Daskalakis, Dick Karp, Elchanan Mossel, Bobak Nazer, Michael Gastpar, June Wang, Jeremy Schiff, David Chu, Dominic Antonelli, Amin Gohari, and Petros Maragos. Petros has been the first researcher I interacted with and my first mentor; I am deeply indebted to him.

Almost everything I know that I did not learn from my collaborators, I learned from some inspired courses. Thanks to Venkat Anantharam for random processes and coding theory, Christos Papadimitriou and Dick Karp for classes on algorithms, David Tse for his wireless class and Alistair Sinclair for his randomized algorithms course.

The Wireless Foundations Lab and Berkeley in general has been an amazing collaboration and social environment. Special thanks to Anand, Bobak, Pulkit, Rahul, Salman, Gwen, Krish, June, Vinod, Dan, Dragan, Wei, and Hari for numerous discussions, not all of them technical. I would like to thank Ruth Gjerde, the calm magical force that solves any administrative problem within seconds and makes grad student life so much easier.

The Bay area Greeks have been the family away from home for me. Thanks to Costas (*The Colocataire*), Alexandra, Maria-Daphne, Nikos, Lefteris, Victoras, Themos and Dora, Antonis and Antonakis, Dimitris, Tasoleris, Manolis, Kostas *D*, Charis and Tasos, Eva, Eleni, Christos and George.

No words can do justice to describe what I owe to my family. I want to thank them for their infinite support over an unbounded time interval and hope that in the future I will be able to give equal love and support.

to my parents

Στους γονεις μου

Αντώνη , που με έμαθε να σκέφτομαι και να ρωτάω.

Ικαρία, που με έμαθε να βλέπω την ομορφιά της ζωής.

Contents

Chapter 1

Introduction

1.1 Outline and Contributions

This thesis consists of two main parts dealing with distributed information storage and distributed processing respectively. Chapters 2 and 3 deal with distributed storage and Chapters 4 and 5 present distributed algorithms for computation and their convergence analysis.

The purpose of distributed storage systems is to store data reliably over long periods of time using a distributed collection of storage nodes which may be individually unreliable. There are several different distributed storage scenarios which include large data centers and peer-to-peer storage systems such as OceanStore [Rhea *et al.*, 2001], Total Recall [Bhagwan *et al.*, 2004], and DHash++ [Dabek *et al.*, 2004], that use nodes across the Internet for distributed file storage. Also it is often desirable to store information (usually obtained from sensing) in wireless sensor networks, creating reliable storage over unreliable motes, for robust data recovery [Dimakis *et al.*, 2006a], especially in catastrophic scenarios [Kamra *et al.*, 2006].

Chapters 2 and 3 deal with finding efficient methods for storing massive amounts

of data over large-scale networks, without centralized coordination. When the storage nodes are individually unreliable (as is the case in sensor networks, peer-to-peer distributed systems, or modern data centers), *redundancy* must be introduced into the system to improve reliability. The simplest form of redundancy is straightforward replication of the data in multiple nodes. It is well known that information representations that use erasure codes require orders of magnitude less redundancy to provide the same level of reliability [Weatherspoon and Kubiatowicz, 2002a] and have been used in numerous applications (e.g. Reed-Solomon codes, Fountain codes). Still most practical large-scale distributed storage systems use replication, for reasons that are related to the problems dealt with in this research. Gmail for example stores 21 copies, a surprisingly high replication factor. Using the simple analysis found in Chapter 3 and assuming a mean time to failure being 1000 days for a typical disk, the use of a $(33, 10)$ erasure code would yield the same availability as 21 replication, but would allow a 600% increase in the amount of data that can be stored. In simple terms, coding mixes information (using linear equations over a field) rather than simply replicating it. This allows a data object *to be spread over more storage nodes without compromising storage efficiency*, by using smaller packet sizes. The benefits of coding for storage are well understood (see e.g. [Weatherspoon and Kubiatowicz, 2002a] and references therein), but there are significant reasons why codes for storage are still not widely used in distributed large-scale systems. The problems of *decentralized encoding* and *code repair* are the ones we address in chapters 2 and 3 respectively, and we hope that the proposed ideas might have an impact in practical distributed storage architectures.

Decentralized codes:

Communication over the network comes with a cost, since bandwidth (related to energy for wireless networks), is a critical resource. Therefore, good storage codes need to be constructed with the minimal possible communication between nodes.

In the terms of codes on graphs, this means that the code should be sparse. Also, in large-scale systems, when events are occurring in multiple distributed locations, global *coordination* is difficult to achieve. Chapter 2 is based on joint work with Vinod Prabhakaran [Dimakis *et al.*, 2005; Dimakis *et al.*, 2006b; Dimakis *et al.*, 2006a] and presents *decentralized erasure codes* which are optimally sparse codes with minimal communication and coordination requirements. Partial recovery of only the most important data has been addressed in follow-up work [Dimakis *et al.*, 2006b; Dimakis *et al.*, 2006a] and [Lin *et al.*, 2007; Wang *et al.*, 2006]. Sparse codes for storage in sensor networks have also been used for time-critical or emergency (e.g. fire, flood, earthquake) scenarios [Kamra *et al.*, 2006] and refinable approximations [Wang *et al.*, 2007]. In our recent book chapter [Dimakis and Ramchandran, 2007] we survey this and related work on encoded information storage in wireless networks.

The repair problem

Chapter 3 deals with the problem of dynamically maintaining an erasure encoded storage system. When the number of storage nodes falls below a threshold due to failures, fresh nodes can be deployed to replace the failed ones and prolong data lifetime. When replication is used, this repair problem is very simple, since a copy of the data object can be obtained from any functioning node. However, when erasure coding is used, the data object (of size $\mathcal{M}$ bits) is divided into k packets and each of the existing storage nodes stores one linear combination of those original packets (of size $\mathcal{M}/k$ bits). In this case, the problem of creating *new encoded packets from encoded packets* is non-trivial. One way of repairing a failure involves communicating k encoded packets, decoding the whole data object, and then generating a new encoded packet using the original data. This approach requires the communication of $\mathcal{M}$ bits in the network, to create only one packet of size $\mathcal{M}/k$. It was commonly believed (see for example [Rodrigues and Liskov, 2005] and references therein) that this k-factor overhead in repair bandwidth is the unavoidable price

required for the added reliability of erasure coding. In fact, all known coding constructions require access to the original data object to generate encoded fragments. In our recent work with Brighten Godfrey and Yunnan Wu [Dimakis *et al.*, 2007; Wu *et al.*, 2007] we show that, surprisingly, erasure codes exist *that can be repaired without communicating the whole data object*. The minimum repair bandwidth problem can be reduced to a network coding problem on an infinite graph and a fundamental tradeoff between storage and bandwidth can be completely characterized. We present information theoretic lower bounds, network codes that achieve every point on this optimal tradeoff, and show how such codes can benefit real storage systems.

Chapters 3 and 4 deal with distributed gossip algorithms that require minimal central knowledge and coordination. Gossip (or average consensus) algorithms, are simple message-passing schemes that solve the problem of computing the average of n sensor measurements in a distributed way. This problem, its connections to the spectrum of stochastic matrices, and mixing of Markov chains has been studied for over thirty years starting with the pioneering work of Tsitsiklis [Tsitsiklis, 1984].

The averaging problem is a useful building block in distributed signal processing, when the goal is to compute a global objective (e.g. the global average of all observations) based on purely local computations. Indeed, an averaging algorithm can be easily converted into a general scheme that computes any linear projection of the sensor measurements. Recently, such algorithms have been proposed for various signal processing tasks like distributed filtering, and distributed detection [Spanos *et al.*, 2005; Saligrama *et al.*, 2006]. Boyd et al. [Boyd *et al.*, 2006] in their influential work, gave a precise characterization of the number of iterations required for gossip algorithms to converge. Our own research started with identifying that even when optimized, nearest neighbor gossip requires $\Theta(n^2)$ messages to estimate the average of n nodes for topologies that are relevant for wireless networks. This quadratic number of messages makes gossip algorithms impractical for energy-limited sensor network sce-

narios. The fundamental reason is that information is diffusing like a random walk, essentially covering distance $\sqrt{h}$ in h hops.

Geographic Gossip

Chapter 3 is based on joint work with Anand Sarwate and analyzes the performance of a novel gossip algorithm, called *geographic gossip* [Dimakis *et al.*, 2006d] that uses geographic information at the nodes to spread gossip in random directions, avoiding diffusive behavior. We show that geographic gossip requires $O(n^{1.5})$ messages to estimate the average, a $\sqrt{n}$-factor decrease in communication over nearest neighbor gossip. Later work used geographic gossip to perform compressed sensing in networks [Rabbat *et al.*, 2006b]. Related recent work relates to exploiting the physical wireless medium [Scaglione, 2007], and performing multi-resolution representations using gossip [Sarkar *et al.*, 2007].

Path Averaging:

The geographic gossip paper left one important open problem unanswered: The algorithm uses location information to greedily route to a randomly selected location in the network, and the initial and final nodes replace their values with their pairwise average. An obvious improvement would allow every node on the routed path to participate in the averaging, at no extra communication cost. While it was clear that this could only improve the convergence time, it was unknown if it would be a constant factor improvement or a change in the $n^{1.5}$ order of magnitude. Chapter 4 is based on joint work performed during the summer of 2006 in EPFL working with Florence Benezit, Martin Vetterli, and Patrick Thiran. The key result of Chapter 4 is that the use of the Poincaré inequality [Diaconis and Stroock, 1991a], a powerful tool for bounding mixing times of Markov chains, can applied to the convergence analysis of path averaging. The result was that geographic gossip with averaging on the routed paths requires only $O(n)$ messages [Benezit *et al.*, 2007], which is order-optimal since it matches obvious lower bounds.

We start with a brief survey of linear erasure codes and network coding. One important point is how the algebraic properties of the generator matrix of a code correspond to requirements from the network algorithm used to construct and maintain the encoded representation.

1.2 Preliminaries on Coding

Erasure coding is a generalization of replication that divides the initial data object into k packets (or blocks) which are then used to generate n encoded packets of the same size. Good erasure codes have the property that *any* k out of the n encoded packets suffice to recover the original k data packets. In erasure problems we assume that the only types of errors that can happen are erasures of packets (which for storage problems correspond to failure of the corresponding storage node), but the packets that survive are always correct. Note also that we will be dealing with erasures of packets, not bits within a packet.

A toy example of storage using a linear code over $GF(2^8)$ is given in Figure 1.1. In the example there are two data nodes X_1 and X_2 and three storage nodes Y_1, Y_2, Y_3. We assume the data nodes have gathered a number of measurements. In the example we choose $u = 8$ bits to represent each number in our field which corresponds to $GF(2^8)$. The bits of the data measurements are divided into blocks of u bits which correspond to elements in $GF(2^8)$ (for example $X_1(1) = 002, X_1(2) = 080, X_1(3) = 220$). The data packet X_1 is routed to storage nodes Y_1, Y_3 and X_2 to Y_2, Y_3. Once a storage node receives one or more data packets, it must select coefficients f_i to multiply the received packets and subsequently add them to construct one encoded packet.

A desired property is that the selection of the coefficients is done without any coordination, i.e. each storage node selects them uniformly and independently in

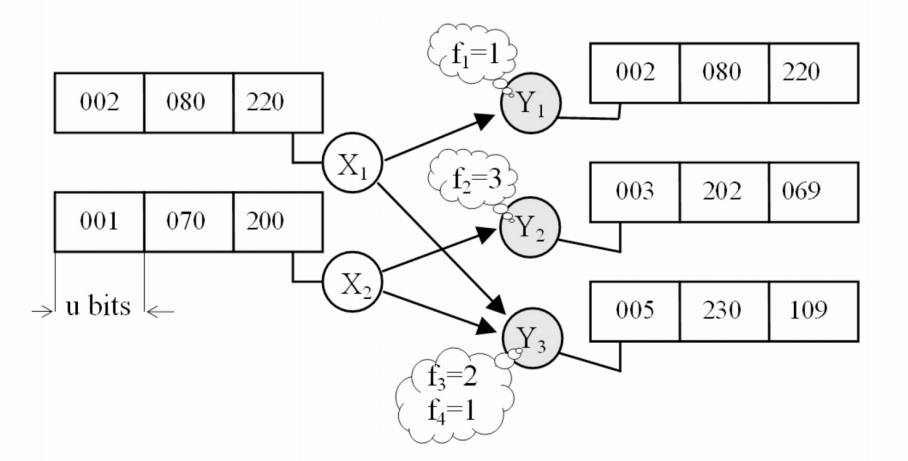


Figure 1.1: A simple example of a linear code over $GF(2^8)$. Here $k = 2$, $n = 3, k/n = 2/3$, $q = 256$. The primitive polynomial of the field is $D^8+D^4+D^3+D^2+1$. Arithmetic is done by representing numbers as binary coefficients of polynomials and doing polynomial operations modulo the primitive polynomial. For example, $70 \times 3 \rightarrow (D^6 + D^2 + D) \times (D + 1) = D^7 + D^6 + D^3 + D \rightarrow 202$.

$GF(2^8)$. Each coefficient then multiplies each block independently, multiple blocks are added (under the arithmetic of the Galois Field) and the results are cascaded into a new block packet Y_i that has exactly the same size as all the data packets. For example Y_3 has stored a packet that corresponds to $2X_1 + 1X_2$. Using this notation we mean that $Y_3(i) = 2X_1(i) + 1X_2(i)$ for $i = 1, 2, 3$. Each storage node will also store the coefficients f_i that it selected. This introduces an overhead storage that can be made arbitrarily small by coding over larger blocks [Ho *et al.*, 2006a; Dimakis *et al.*, 2006a].

Notice that in Figure 1.1 any two out of the three encoding packets can be used to reconstruct the original data.

In general, linear codes can be represented using their *generator matrix* in the form

$$s = mG, \tag{1.1}$$

where s is an $1 \times n$ encoded vector that is stored, m is $1 \times k$ data vector and G is a $k \times n$ matrix with elements selected from a field $GF(q)$. For the example in Figure 1.1,

$$G = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 3 & 1 \end{pmatrix}. \tag{1.2}$$

To reconstruct m the receiver must invert a $k \times k$ sub-matrix G' of G. The key property required for successful decoding (that is also true for the example) is that any sub-matrix selection of G' forms a *full rank matrix*. When this is the case, decoding corresponds to solving a system of linear equations over $GF(q)$ (using for example, Gaussian elimination). It is well known that good erasure codes can yield much higher reliability compared to replication schemes for the same number of storage nodes, and this increased reliability is the main reason codes are attractive for storage applications.

A matrix that has the property that *all* the square sub-matrices G' are full rank corresponds to an Maximum Distance Separable (MDS) code and such combinatorial constructions are quite difficult to achieve.

Reed-Solomon codes [Reed and Solomon, 1960] construct such matrices by exploiting properties of polynomials over finite fields. The key idea is that any k interpolation points suffice to recover the coefficients of a degree $k-1$ polynomial. The smallest field size q for which MDS codes exist is unknown, and related to the MDS conjecture of algebraic coding theory:
(MDS Conjecture) Let G be a $k \times n$ matrix over $GF(q)$ such that every square sub-matrix G' is nonsingular. Then $q + 1 \geq n + k$.

A relaxation of this requirement (nearly-MDS) is that *almost all* the square sub-matrices G' are full rank, or equivalently that a randomly selected G' will be full rank with high probability. A *random linear code* over $GF(q)$ is the code generated

by a matrix G that has each entry selected uniformly and independently from the finite field. It is well known [Acedanski *et al.*, 2005] that the probability of a randomly selected G' being full rank can made arbitrarily close to one, by selecting a sufficiently large field size q. Reed-Solomon codes, are widely employed in numerous applications like computer network distributed storage systems, and redundant disk arrays. Low-density parity-check (LDPC) codes and more recently Fountain codes [Luby, 2002] were proposed as alternatives with randomized construction and faster encoding and decoding times.

1.2.1 Network Coding

Network coding is an exciting new paradigm for communication in networks where data packets are treated as entities which can be algebraically combined rather than simply routed and stored. The first major result [Ahlswede *et al.*, 2000a] was a generalization of the max-flow min-cut theorem for multicasting. If there is one single source and multiple receivers, each receiver cannot hope to have throughput higher than the minimum cut separating it from the source, even if it was the only node being served. The theorem of Ahlswede et al. [Ahlswede *et al.*, 2000a] states that if coding in the intermediate nodes of the network is allowed, *all the receivers* can have throughput equal to the minimum of the min-cuts separating each one from the source. In other words all the receivers can have the same throughput as the one with the weakest connection to the source, without limiting each other. It is easy to construct examples where such throughput cannot be achieved by simply routing packets from the source to the receivers. Subsequently it was shown that linear coding suffices to achieve the multicast capacity [Li *et al.*, 2003; Koetter and Médard, 2003] and that random linear coding at intermediate nodes will suffice with high probability [Ho *et al.*, 2006a] for sufficiently large field size.

While most of the initial research on network coding focused on multicasting throughput, the fundamental idea of coding in intermediate nodes in networks has been shown to have advantages in other scenarios such as minimizing network resources [Lun *et al.*, 2006], network diagnosis [Wu and Li, 2006] and communication in wireless networks [Katti *et al.*, 2006; Petrović *et al.*, 2006; Wu, 2006]. See also [Fragouli *et al.*, 2006] for a general introduction and other applications of network coding.

Chapter 2

Decentralized Erasure Codes

2.1 Introduction

In this chapter we will study the problem of creating an erasure code for storing information in multiple storage devices that are individually unreliable, and connected in a network. As an application consider a sensor network deployment in a remote and inaccessible environment where sensor nodes are taking measurements (possibly after processing) and storing data in the network, over long time periods. A data collector may appear at any location in the network and try to retrieve as much useful data as possible. Another scenario is a sensor network deployed in a time-critical or emergency situation (e.g. fire, flood, earthquake). Here, the focus is on maximizing the amount of sensed data than can be retrieved from a rapidly failing network. In both scenarios, many storage nodes are expected to fail and redundancy is necessary to guarantee the required reliability. This redundancy in the information representation can be introduced either through replication or through erasure coding. It is well known that information representations that use erasure codes require far less redundancy to provide the same level of reliability [Weatherspoon and Kubiatowicz,

2002a] and many storage schemes use erasure coding techniques, often based on Reed-Solomon codes [Reed and Solomon, 1960]. After extensive studies, essentially optimal erasure codes exist today, with linear encoding and decoding complexity [Luby *et al.*, 2001; Shokrollahi, 2006]. However, for applications that require large-scale distributed storage, over unstructured (and possibly dynamic) networks, new issues arise that have not been addressed in the classical coding theory literature: Specifically:

- *Communication* between storage and data nodes comes with a cost, since energy is a precious resource in sensor networks. Therefore, the code should be constructed with the minimal possible communication between nodes. This means that sparsity in the generator matrix of the code is critical for such applications.

- The information is sensed in multiple distributed locations and global *coordination* is difficult to achieve. Hence the code construction should be distributed and based on local knowledge.

- The sensor network will be deployed in a dynamic environment and the the encoded storage might need to *evolve over time*, to reflect such dynamic changes. For example when storage nodes are failing, new encoded packets need to be generated from existing encoded packets, naturally leading to network coding schemes.

In this chapter we discuss these issues and various related schemes that have been proposed in the recent literature including our own contributions. In summary, we will be interested in distributed, scalable and energy-efficient algorithms to generate and dynamically maintain encoded information representations in networks.

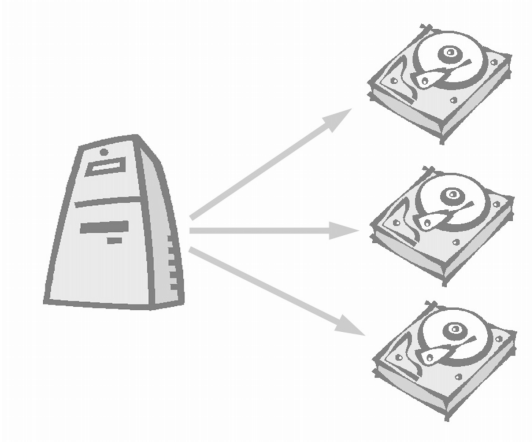


Figure 2.1: The classical distributed storage setup. Data which is initially centralized is encoded and stored in distributed storage nodes.

2.1.1 The Distributed Networked Storage Problem

We will be using the abstractions of a *data node* which is a *source* of information that must be stored, and a *storage node* which corresponds to a storage device with limited memory and communication capabilities. A physical sensor mote can have both sensing capabilities and sufficient memory, and hence can be both a data node and a storage node of our abstract model. This separation is useful because it simplifies the presentation and can be easily mapped back to actual devices.

The classical distributed storage problem consists of having multiple (distributed) storage nodes (e.g. hard disks) for storing data which is initially located at *one single data node* (see Figure 2.1).

The *distributed networked storage* problem arises when both the data sources and the storage nodes are distributed (see Figure 2.2) and hence we have multiple data and storage nodes.

We make the following assumptions:

- We assume that there are k data-generating nodes and without loss of generality we will assume that each data node generates one data packet containing the

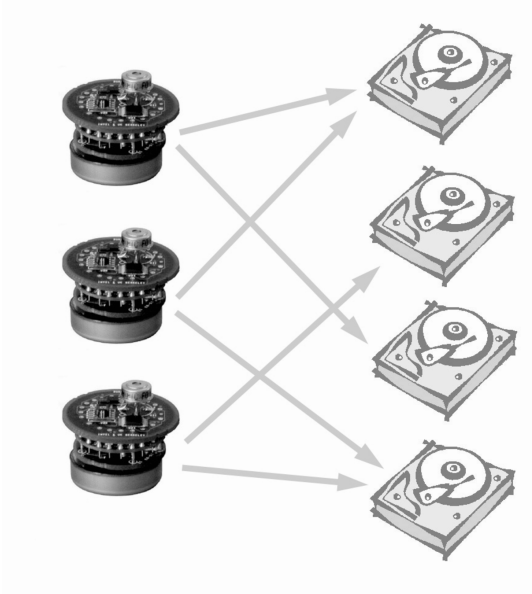


Figure 2.2: Distributed networked storage. Initially distributed data stored in multiple storage nodes.

information of interest.

- Further, assume we have $n \geq k$ storage nodes that will be used as storage and relay devices. Sensor nodes have limited memory, and we model that by assuming that each node can store only one data packet (or a combination having the same number of bits as a data packet). This is a key requirement to the scalability of the network. A data packet contains measurements over a time interval and can have significant size.

- The ratio $k/n = R$ (code rate) is assumed fixed as k and n scale. For example, we can assume that some fixed ratio (for example 10%) of nodes in a sensor network are generating data. These assumptions are only to simplify the presentation, and in practice the k data nodes and n storage nodes can be any arbitrary (possibly overlapping) subsets of nodes in a larger network.

- We are interested in schemes that require no routing tables, centralized pro-

cessing or global knowledge or coordination of any sort. We rely on a packet routing layer that can route packets to *uniformly random locations* in the network. Constructing such *random sampling algorithms* which are distributed and localized is key for the construction of codes in networks.

2.2 Decentralized Erasure Codes

When trying to store linear combinations of data as information representations in sensor networks, new issues arise that make the existing codes unsuitable. Both random linear codes and Reed-Solomon codes have generator matrices that are *dense*, i.e. almost all the entries of G are non-zero. That means that every data node needs to send its packet to almost all n storage nodes to create the code generating $\Theta(n^2)$ communicating pairs (since $k = Rn$). A second desirable property is that the code can be created without coordination, and more specifically that each data node is choosing where to route its packet independently and also that the storage nodes are selecting their coefficients independently. Algebraically, this corresponds to having a code where every row of the generator matrix is created independently and is sparse. A code with this row independence property is called *decentralized* [Dimakis *et al.*, 2006a] and this property leads to stateless randomized network algorithms to generate the encoded information.

We want to create an information representation that has the property that a data collector can query *any k storage nodes and use the results to reconstruct the original k packets* (with high probability). For instance, a data collector can get this data out of some k neighboring storage nodes in its immediate vicinity to minimize the latency in a sensor network scenario. Finally we assume that the data collector has enough memory to store k packets and enough processing power to run the decoding algorithm, which as we will show corresponds to solving a (sparse) system of k linear

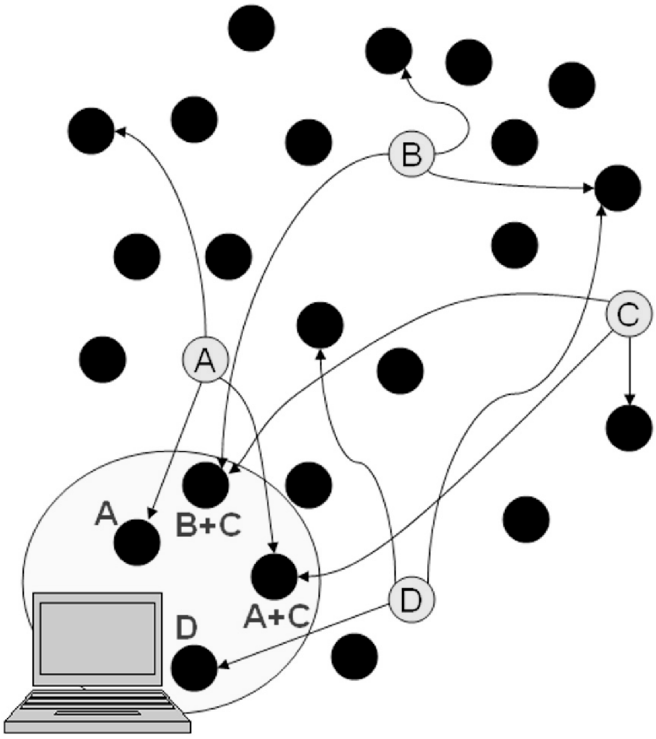


Figure 2.3: Example of using linear codes for distributed storage. In this example there are $k = 4$ data nodes measuring information that is distributed and $n = 23$ storage nodes. We would like to diffuse the data to the storage nodes so that by accessing any 4 storage nodes it is possible to retrieve the data. Each data node is pre-routing to 3 randomly selected storage nodes. Each storage node has memory to store only one data packet so the ones who receive more than one packet store a linear combination of what they have received. The data collector in the example can recover the data by having access to (A, B+C, A+C, D).

equations in a finite field.

Decentralized erasure codes have minimal data node degree which corresponds to a maximal sparsity of the generator matrix and minimal number of pre-routed packets. Note however, that we do not claim optimality of the distributed networked storage system as a whole. This is because we rely on a packet routing layer instead of jointly optimizing across all network layers.

Decentralized erasure codes are random linear codes over a finite field F_q with a specific randomized structure on their generator matrix. Each data packet D_i is seen as a vector of elements of a finite field f_i . We denote the set of data nodes by V_1 with $|V_1| = k$ and storage nodes by V_2, $|V_2| = n$. We will now give a description of a randomized construction of a bipartite graph that corresponds to the creation of a decentralized erasure code. Every data node $i \in V_1$ is assigned a random set of storage nodes $N(i)$. This set is created as follows: a storage node is selected uniformly and independently from V_2 and added in $N(i)$ and this procedure is repeated $d(k)$ times. Therefore $N(i)$ will be smaller than $d(k)$ if the same storage node is selected twice. In fact, the size of the set $N(i)$ is exactly the number of coupons a coupon collector would have after purchasing $d(k)$ coupons from a set of n coupons. It is not hard to see that when $d(k) \ll n$, $N(i)$ will be approximately equal to $d(k)$ with high probability.

Denote by $N(j) = \{i \in V_1 : j \in N(i)\}$ the set of data nodes that connect to a storage node. Each storage node will create a random linear combination of the data nodes it is connected with:

$$S_j = \sum_{\forall i : i \in N(j)} f_{ij} D_i \tag{2.1}$$

where the coefficients f_{ij} are selected uniformly and independently from a finite field F_q. Each storage node also stores the f_{ij} coefficients, which requires an overhead storage of $N(j)(\log_2(q) + \log_2(k))$ bits.

This construction can be summarized into

$$s = mG \tag{2.2}$$

where s is a $1 \times n$ vector of stored data, m is $1 \times k$ data vector and G is a $k \times n$ matrix with non-zero entries corresponding to the adjacency matrix of the random bipartite graph we described. The key property that allows the decentralized construction of the code is that *each data node is choosing its neighbors independently and uniformly* or equivalently, *every row of the generator matrix is created independently* and has $N(i) = O(d(k))$ nonzero elements. The decentralized property (row independence), was proposed in our work [Dimakis *et al.*, 2005] and independently in [Acedanski *et al.*, 2005] and leads to stateless robust randomized algorithms for distributed networked storage. We compare our results with random linear coding for distributed networked storage proposed in [Acedanski *et al.*, 2005] in section 2.3.4.

A data collector querying k storage nodes will gain access to k encoded packets. To reconstruct, the data collector must invert a $k \times k$ submatrix G' of G. Therefore, the key property required for successful decoding is that any selection of G' forms a *full rank matrix* with high probability. Clearly $d(k)$ is measuring the sparsity of G. Making $d(k)$ as small as possible is important since it is directly related with overhead storage, decoding complexity and communication cost. Our main contribution is identifying how small $d(k)$ can be made for matrices that have the decentralized property to ensure that their submatrices are full rank with high probability. The main results of this chapter are the following theorems:

Theorem 1. *Let G be a random matrix with independent rows constructed as described. Then, $d(k) = c\ln(k)$ is sufficient for a random $k \times k$ submatrix G' of G to*

be nonsingular with high probability. More specifically,

$$Pr[det(G') = 0] \leq \frac{k}{q} + o(1), \tag{2.3}$$

for any $c > 5\frac{n}{k}$.

Theorem 2. *(Converse) If each row of* G *is generated independently (Decentralized property), at least* $d(k) = \Omega(\ln(k))$ *is necessary to have* G' *invertible with high probability.*

From the two theorems it follows that $d(k) = c\ln(k)$ is (order) optimal, therefore, decentralized erasure codes have minimal data node degree and logarithmically many nonzero elements in every row.

Decentralized erasure codes can be decoded using Maximum Likelihood (ML) decoding, which corresponds to solving a linear system of k equations in $GF(q)$. This has a decoding complexity of $O(k^3)$. Note however that one can use the sparsity of the linear equations and have faster decoding. Using the Wiedemann algorithm [Wiedemann, 1986] one can decode in $O(k^2 \log(k))$ time on average with negligible extra memory requirements.

2.3 Analysis and Proofs

Proof of Theorem 1

To establish that decentralized erasure codes will be decodable, we need to show that a randomly selected square submatrix G' is full rank with high probablity. For this proof we rely heavily on the Theorem 3 and use techniques similar with the ones used by Ho et al. [Ho *et al.*, 2006a].

It suffices to show:

$$\det G' \neq 0.$$

We will use the concept of a perfect matching: a bipartite graph will have a perfect matching (P.M.) if there exists a subset $E' \subseteq E$ of its edges so that no two edges in E' share a common vertex and all the vertices connect to an edge in E'. There is a close connection between determinants of matrices and graph matchings which for the bipartite case is given by Edmonds' Theorem [Motwani and Raghavan, 1995a]. By construction, every row of G' has a logarithmic number of non-zero coefficients chosen uniformly and independently from a finite field F_q. Denote these coefficients by $f_1, f_2, \cdots f_L$. Their actual number L is random and approximately equal (and in fact, smaller than) $ck\ln(k)$. It suffices to show that the determinant of G' is nonzero w.h.p. Note that

$$\det(G') = \sum_{\pi} sgn(\pi) \prod_{i=1}^{k} g'_{i,\pi(i)} \tag{2.4}$$

where we are summing over all the permutations of $\{1, 2, \cdots k\}$ and $g'_{i,j}$ is the i, jth element of G'. Notice that this is a multivariate polynomial

$$\det(G') = P(f_1, f_2, \cdots, f_L).$$

There are two fundamentally different cases for the determinant to be zero. If for each term corresponding to each permutation there existed one or more zero elements then the determinant would be identically zero (not a function of $f_1, f_2, \cdots f_L$). Now the key step is to notice that each permutation corresponds to exactly one potential matching of the bipartite graph. Therefore, the graph has a perfect matching if and only if $\det(G')$ is not identically zero (Edmonds' Theorem [Motwani and Raghavan, 1995a]). Theorem 3 establishes exactly that the random bipartite graphs we construct have perfect matchings. The other case is when $deg(G')$ is a non-zero polynomial but the specific choices of $f_1, f_2, \cdots f_L$ correspond to one of its roots. It is clear that this is a rare event and we can bound its probability using the Schwartz-Zippel Theorem

[Motwani and Raghavan, 1995a]. Notice that the degree of $\det(G')$ is exactly k when there exists a perfect matching so we obtain a bound on the probability of failure conditioned on the existence of a perfect matching:

$$Pr(\det(G') = 0 | \det(G') \not\equiv 0) \leq \frac{k}{q}.$$

Which leads us to

$$Pr(\det(G') = 0) \leq Pr(\det(G') \equiv 0) + \frac{k}{q}(1 - Pr(\det(G') \equiv 0)). \tag{2.5}$$

By Theorem 3, $Pr(\det(G') \equiv 0) = o(1)$ therefore

$$Pr(\det(G') = 0) \leq k/q + o(1). \tag{2.6}$$

■

Proof of Theorem 2 (Converse) It is a standard result in balls and bins analysis [Motwani and Raghavan, 1995a] that in order to cover n bins w.h.p. one needs to throw $\Theta(n \ln n)$ balls (See also case III in proof of Th. 3). Notice that in our case, covering all the storage nodes is necessary to have a full rank determinant (since not covering one corresponds to having a zero column in G). Therefore any scheme that has data nodes acting independently and uniformly will require at least $\Omega(\ln k)$ connections per data node, just to ensure that the storage nodes are covered.

■

We have therefore demonstrated that the key technical condition we need to prove is that the random bipartite graphs we construct have a perfect matching [Bollobás, 2000] with high probability. The existence of a perfect matching guarantees that the max flow that can go through the network is sufficient. Our theoretical contribution, which may be of independent interest, is in quantifying how sparse these random

bipartite graphs can be under these constraints. The proof is obtained by using an extension of a combinatorial counting technique introduced by P. Erdős and A. Rényi in [Erdös and Sachs, 1963; Bollobás, 2001] for analyzing matchings in random bipartite graphs. The extension stems from the dependencies on the data nodes which destroy the symmetry assumed in [Erdös and Sachs, 1963; Bollobás, 2001] thereby complicating matters.

We define the graph $B_{\ln k-left-out}$ as the random bipartite graph with two sets of vertices, V_1, V_2, where $|V_1| = k$, $|V_2| = n$, $n = \alpha k, (\alpha > 1)$. Every vertex in V_1 connects with $c\ln(k)$ vertices of V_2 each one chosen independently and uniformly with replacement. If two edges connect the same two vertices we identify them. Then we pick a subset $V_2' \subset V_2$ where $|V_2'| = k$ and form the random bipartite graph $B'_{\ln k-left-out} = |V_1| \cup |V_2'|$. Edges that connect to $V_2 \setminus V_2'$ are deleted.

This graph corresponds to the submatrix G' and we need to establish that $B'_{\ln k-left-out}$ has a perfect matching w.h.p.

Theorem 3: *Let $B'_{\ln k-left-out}$ be a bipartite graph with $|V_1| = |V_2'| = k$ obtained from $B_{\ln k-left-out}$ by taking a random subset of k storage nodes. $B'_{\ln k-left-out}$ has a perfect matching with probability $1 - o(1)$ as $k \to \infty$.*

Proof: For a set of nodes $A \subset V_i$ of a bipartite graph B, we denote $\Gamma(A)$=$\{y$: $xy \in E(B)$for some $x \in A\}$. So $\Gamma(A)$ is simply the set of nodes that connect to nodes in A.

A key result used in this proof is Hall's Theorem. We use it in the following form (which is easily derived from the standard Theorem [Bollobás, 2000; Bollobás, 2001]):

Lemma 1. *Let B be a bipartite graph with vertex classes V_1,V_2' and $|V_1| = |V_2'| = k$. If B has no isolated vertices and no perfect matching, then there exists a set $A \subset V_i$ $(i = 1, 2)$ such that:*
i) $|\Gamma(A)| = |A| - 1$

ii) The subgraph $A \cup \Gamma(A)$ is connected

iii) $2 \leq |A| \leq (k+1)/2$.

The event that B has no perfect matching can be written as the union of two events. Specifically, let E_0 denote the event that B has one or more isolated vertices: $P(\text{B has no P.M.}) = P(E_0 \bigcup \exists A)$ (for some set A satisfying Lemma (1)) Therefore by a union bound we have:

$$P(\text{B has no P.M.}) \leq P(E_0) + P(\exists A).$$

We will treat the isolated nodes event later. We know from Lemma (1) that the size of A can vary from 2 to $(k+1)/2$, so we obtain the union bound:

$$P(\exists A) = P(\bigcup_{i=2}^{(k+1)/2} (\exists A, |A| = i)) \leq \sum_{i=2}^{(k+1)/2} P(\exists A, |A| = i). \tag{2.7}$$

We can further partition into two cases, that the set A belongs to V_1 (the data nodes) or V_2' (the k storage nodes used to decode).

$$P(\exists A) \leq \sum_{i=2}^{(k+1)/2} P(\exists A \subset V_1, |A| = i) + P(\exists A \subset V_2', |A| = i) \tag{2.8}$$

So we now bound the probabilities $P(\exists A \subset V_1, |A| = i)$ and $P(\exists A \subset V_2', |A| = i)$ using a combinatorial argument. *Case I: A belongs in the data nodes:* Suppose we fix i nodes $A_1 \subset V_1$ and $i-1$ nodes on $A_2 \subset V_2'$. Then the probability that a set $A = A_1$ satisfies the conditions of lemma (1) with $\Gamma(A) = A_2$ is equal to the probability that all the edges starting from A_1 will end in A_2 or are deleted. Note however that every node in V_1 picks $c\ln(k)$ neighbors from the set V_2 (which is the large set of $n = \alpha k$ nodes). We bound the probability by allowing all edges starting from A_1 to land in $A_2 \cup V_2 \setminus V_2'$. Therefore we have $ci\ln(k)$ edges that must land in $A_2 \cup V_2 \setminus V_2'$ and

$|A_2 \cup V_2 \setminus V_2'| = i - 1 + (\alpha - 1)k$. Note that all the other edges can land anywhere and that would not affect $|\Gamma(A)|$. Therefore, since there are $\binom{k}{i}$ choices for A_1 and $\binom{k}{i-1}$ choices for A_2 we have:

$$P(\exists A \subset V_1) \leq \sum_{i=2}^{(k+1)/2} \binom{k}{i}\binom{k}{i-1}\left(\frac{i-1+(\alpha-1)k}{\alpha k}\right)^{ci\ln(k)} \tag{2.9}$$

We can always bound this sum by its maximum value times k (since there are fewer than k positive quantities added up). Therefore it suffices to show that

$$kP(\exists A \subset V_1, |A| = i) = o(1), \qquad \forall i \in [2, (k+1)/2] \tag{2.10}$$

as $k \to \infty$.

From Stirling's approximation we obtain the bound [Bollobás, 2001]

$$\binom{k}{i} \leq (\frac{ek}{i})^i$$

and also it is easy to see that $(\frac{ek}{i-1})^{i-1} \leq (\frac{ek}{i})^i$ when $i \leq k$.

If we denote

$$\xi = \left(\frac{i-1+(\alpha-1)k}{\alpha k}\right) < 1$$

and use these two bounds we obtain :

$$P(\exists A \subset V_1, |A| = i) \leq \exp\Big(\ln(k)(2i + ic\ln(\xi)) + 2i(1 - ln(i))\Big). \tag{2.11}$$

If we multiply by k we get from (2.10) that it suffices to show

$$\exp\Big(\ln(k)(2i + ic\ln(\xi) + 1) + 2i(1 - ln(i))\Big) = o(1), \tag{2.12}$$

for all $i \in [2, (k+1)/2]$, as $k \to \infty$. Therefore, for this exponential to vanish it is sufficient to have the coefficient of $\ln k$ be negative:

$$2i + ic\ln(\xi) + 1 < 0, \tag{2.13}$$

which gives us a bound for c:

$$c > \frac{-(1+2i)}{i\ln(\xi)}. \tag{2.14}$$

Notice that $\xi < 1$ and therefore it is possible to satisfy this inequality for positive c. This bound should be true for every $i \in [2, (k+1)/2]$. So using

$$\frac{1+2i}{i} = \frac{1}{i} + 2 \leq \frac{5}{2}, \tag{2.15}$$

and

$$\xi = \frac{i-1+(\alpha-1)k}{\alpha k} \leq \frac{(k+1)/2+(\alpha-1)k}{\alpha k} \approx \frac{\alpha - 1/2}{\alpha}, \tag{2.16}$$

$$\frac{-1}{\ln(\xi)} \leq \frac{-1}{\ln(\frac{\alpha-1/2}{\alpha})}. \tag{2.17}$$

Therefore, a sufficient condition for $P(\exists A \subset V_1)$ to vanish is

$$c > \frac{-5}{2\ln(\frac{\alpha-1/2}{\alpha})} \simeq 5\alpha. \tag{2.18}$$

Case II: A belongs in the storage nodes: With the same technique, we obtain a bound if the set A is on the data nodes. This time we pick $A \subset V_2'$ with $|A| = i$ and we want $|\Gamma(A)| = i - 1$. So we require that all edges that connect to A end in a specific set $A_2 \in V_1$. The extra requirement that $A \cup \Gamma(A)$ should be connected, further reduces the probability and is bounded away. To have $|\Gamma(A)| = A_2$, it must be the case that all the edges that start from $V_1 \setminus A_2$ land outside A. There are

$c(k-(i-1))\ln(k)$ such edges and each one lands outside A with probability $\frac{\alpha k-i}{\alpha k}$. We therefore obtain the bound:

$$P(\exists A \subset V_2, |A| = i) \leq \binom{k}{i-1}\binom{k}{i}\left(\frac{\alpha k - i}{\alpha k}\right)^{c(k-(i-1))\ln(k)}, \tag{2.19}$$

which yields the condition for c :

$$c > \alpha\frac{k}{i}\frac{2i+1}{k-i} = 2\alpha\frac{k}{k-i} + \alpha\frac{k}{i(k-i)} \tag{2.20}$$

Now notice that this is a convex function of i so the maximum is obtained at $i = 2$ or $i = \frac{k+1}{2}$. By substituting $i = 2$ and $i = \frac{k+1}{2}$ we find that these inequalities are always dominated by (2.18). So finally we require that $c > 5\alpha$.

Case III: There exist no isolated nodes: We will say that a data or storage node is isolated when it connects to no storage or data node respectively. Bounding the probability of this event $P(E_0)$ is easier to deal with. Notice that data nodes cannot be isolated by construction. The αk storage nodes receive totally $kc\ln(k)$ independent connections and we need to show that they are all covered by at least one data node w.h.p. Using a standard bound we obtain the following result ([Motwani and Raghavan, 1995a]):

Let C denote the number of connections required to cover all αk data nodes. then

$$P[C > \beta\alpha k\ln(\alpha k)] \leq (\alpha k)^{-(\beta-1)}, \tag{2.21}$$

which shows that any $\beta > 1$ (we require $\beta > 5$) will suffice to cover all the data nodes with high probability.

Therefore from combining all the required bounds for c we find that $c > 5\alpha = 5\frac{n}{k}$ is sufficient for the bipartite graph to have a perfect matching with high probability.

■

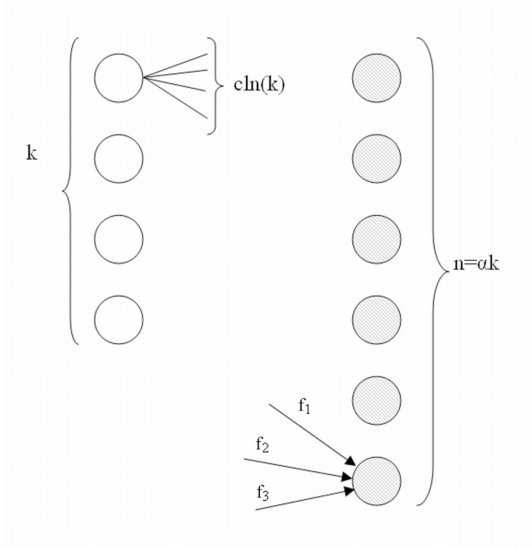


Figure 2.4: Decentralized erasure codes construction. There are $d(k) = c\ln(k)$ edges starting from each data node and landing independently and uniformly on the storage nodes.

2.3.0.1 Randomized Network Algorithm

There is a very simple, robust randomized algorithm to construct a decentralized erasure code in a network: Each data node picks one out of the n storage nodes randomly and routes its packet to a randomly selected storage node. By repeating this process $d(k) = c\ln(k)$ times, we construct the decentralized erasure code. Note that we require a network layer mechanism that can route packets to randomly selected storage nodes in the network. Having a simple distributed mechanism that can perform this task with localized knowledge is key for many randomized algorithms and we discuss fast distributed algorithms for sampling nodes in Chapter 4. Each storage node multiplies (over the finite field) whatever it happens to receive with coefficients selected uniformly and independently in $F(q)$ and stores the result and the coefficients. A schematic representation of this is given in Figure 2.4.

2.3.0.2 Storage Overhead

In addition to storing the linear combination of the received data packets, each storage node must also store the randomly selected coefficients f_i. The number of coefficients can be bounded by the number of balls that land into a bin when throwing $ck\ln(k)$ balls into n bins. It is standard problem in probabilistic analysis of algorithms [Motwani and Raghavan, 1995a] that the maximum load (the maximum number of coefficients a storage node will have to store) is $O(\log(k))$ with probability at least $1 - o(1)$. The total number of overhead bits to store the coefficients and data packet IDs is $O(\log(k)(\log(q)+\log(k)))$ which can be easily made negligible by picking larger data packet sizes. Notice that if we denote by $u = \log_2(q)$ the number of bits required to store each f_i, one can reduce the probability of error exponentially in the overhead bits.

2.3.0.3 Connections to network coding

An equivalent way of thinking of the distributed networked storage problem is that of a random bipartite graph connecting the k data nodes with the n storage nodes and then 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 independently in [Dimakis *et al.*, 2005; Jiang, 2006].

It has been shown that random linear network codes [Li *et al.*, 2003; Ho *et al.*, 2006a] 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 [Ho *et al.*, 2006a] on the (random) bipartite graph connecting the data and the storage nodes, where each edge corresponds to one routed

packet. One key property is that in distributed storage, the communication graph does not correspond to any physical links but to virtual routing selections that are made by the randomized algorithm. Therefore this graph is not given, but can be *explicitly designed to minimize communication cost*. Essentially, we are trying to make this random bipartite graph as sparse as possible, while keeping the flow high enough and also allowing each data node to act independently. All good sparse-graph codes have the property that they have few edges ($o(n^2)$) connecting the data nodes and the storage nodes but can still guarantee very good connectivity between the any two subsets. Such bipartite graphs are called expanders [Alon and Spencer, 2000] and are fundamental combinatorial objects for coding theory. It is easy to show that if one requires all $\binom{n}{k}$ data collectors to have k-connectivity with the data nodes, the corresponding bipartite graph needs to be dense. It is the probabilistic relaxation (a random data collector will have k-connectivity with high probability) that makes sparsity possible. This concept leads to *probabilistic expanders* that are formally defined and used for error correction in [Daskalakis *et al.*, 2007].

2.3.1 Fountain Codes

Fountain codes [Luby, 2002; Shokrollahi, 2006] are linear codes over $GF(2)$ with sparse generator matrices and fast encoding and decoding algorithms. In particular, for LT codes [Luby, 2002], each encoded packet is created by first selecting a degree d from a carefully designed degree distribution (called the *robust soliton* [Luby, 2002]), and then taking the bitwise XOR of d randomly selected data packets. Therefore, fountain codes have the *rateless property*: every encoded packet is generated independently and there exists no predetermined rate since they can potentially generate an unbounded number of encoded packets. This corresponds to having every *column* of the generator matrix being independent and sparse (with logarithmic average degree similar to the

decentralized codes). The degree distribution of the encoded packets is carefully designed so that a data collector who collects $k+\epsilon$ random packets (where the overhead ϵ is asymptotically vanishing for large k) can decode with a fast back-substitution algorithm which is special case of belief propagation [Luby *et al.*, 2001]. Raptor codes [Shokrollahi, 2006] manage to reduce the degrees from logarithmic to constant by using an appropriate pre-code. This idea cannot be used for the distributed storage problem, since constructing the pre-code would require centralized processing.

In this context, one can think of the decentralized property as being the *transpose* of the rateless property. This is because in decentralized codes, it is the rows of the generator matrix that are independent and this corresponds to having each data node acting independently. For sensor network applications, one implicit assumption is that it is easier for a data node to send its data to $d(k)$ randomly selected storage nodes than it is for a storage node to find and request packets from $d'(k)$ data nodes. This is true for many practical scenarios in which there are fewer data nodes that might also be duty-cycled or failing.

2.3.2 Partial data recovery

So far we have been addressing the problem of recovering all k data packets by querying k storage nodes. For this scenario, fountain codes are harder to create in networks, since creating the robust soliton degree distribution at storage nodes requires data node coordination. They however have the advantage of smaller field size (only binary operations) and lower computational complexity at the decoder ($O(k \log k)$ for LT codes versus $O(k^2 \log k)$ for decentralized codes). The pre-coding idea of Raptor codes cannot be easily performed over a network because it requires centralized processing.

Fountain codes can be used for partial recovery problems, where one is interested

in querying fewer than k nodes and recover partial information. Creating a fountain code over a network where the data nodes are randomly located on a grid has been addressed in [Dimakis *et al.*, 2006b]. In this work there is no pre-code, and the user is interested in recovering $(1-\delta)k$ data packets by querying $(1+\epsilon)k$ storage nodes. Random walks [Lin *et al.*, 2007] can be used to create fountain encoded packets in sensor networks, to guarantee the persistence and reliability of cached data.

Sanghavi [Sanghavi, 2007] investigated the optimal degree distribution for fountain codes when one is interested in recovering $(1-\delta)k$ data packets. Upper bounds on the performance of any degree distribution and lower bounds achieved by optimized distributions for any δ are presented in [Sanghavi, 2007].

2.3.3 Growth codes

In sensor network applications involving catastrophic or emergency scenarios such as floods, fires, earthquakes etc., the queries need to be adjusted to network dynamics. The setup is a rapidly failing sensor network where some nodes are sensing information that needs to reach the data collectors as soon as possible. Kamra et al. [Kamra *et al.*, 2006] show how fountain codes can be used for such applications and how the degree distribution needs to evolve over time to maximize the number of immediately recoverable data packets. Specifically, the authors design a dynamically varying degree distribution for partial network recovery to adapt to the data collector having received some data packets already and maximize the probability that the next packet is useful immediately. Growth codes initially create uncoded packets (since a data collector will have received nothing at the time and only degree one packets can be immediately useful). The degree distribution switches to pairwise XORs when the probability that a data collector already has a randomly selected packet becomes larger than the probability that the XOR cannot be decoded immediately.

2.3.4 Comparison with Random Linear Coding

In [Acedanski *et al.*, 2005] the authors propose the use of random linear coding inspired by network coding for distributed networked storage with one centralized server and multiple storage locations. They compare traditional erasure codes, uncoded storage and random linear coding motivated by network coding, and demonstrate that there are significant gains in using random linear coding. In random linear coding, every element in the generator matrix of the code is selected independently and uniformly from a finite field F_q. This corresponds to matrices that are dense since they have a constant fraction of nonzero elements, essentially having $d(k) = \Theta(k)$.

The main difference between our work and [Acedanski *et al.*, 2005] is that we address the problem of having multiple distributed sources and no centralized server. Further, we identify how sparse can the generator matrices of decentralized codes can be, and give a simple randomized way of constructing them in a network. Sparsity leads to smaller overhead storage and more importantly, reduced communication and decoding complexity.

Specifically, random linear coding requires an overhead storage space of $O(k \log(q))$ bits, while decentralized erasure codes only $O(\log(k)(\log(q)) + \log(k)))$. The overhead storage costs are usually small if one codes over large data packets hence the communication and complexity gains are more important. If one were to use random linear coding for the multiple source networked storage problem, each data node would have to send its data to $O(n)$ storage nodes, and the total cost would be the same as flooding all the information everywhere. However using decentralized erasure codes each data node has to communicate with only $O(\log(k))$ storage nodes. As far as decoding complexity is concerned, random linear coding requires $O(k^3)$ operations to invert a dense matrix, while decentralized erasure codes can be decoded in $O(k^2 \log(k))$ by exploiting sparsity [Wiedemann, 1986].

2.4 Sensor Network Scenarios

In this section we show how decentralized erasure codes can be applied to various sensor network scenarios and analyze their performance. It is important to realize that one can pick the k data nodes and the n storage nodes to be *any arbitrary subsets of nodes* of a larger network. The exact choices depend on the specific sensing application. The only requirement that we impose is that n/k should remain fixed as the network scales.

In general, it is easy to determine the total communication cost involved in creating a decentralized erasure code. Each data node pre-routes to $5\frac{n}{k}\ln k$ storage nodes, therefore the total number of packets sent will be $5n\ln k$. To determine the communication cost in terms of radio transmissions, we need to impose a specific network model for routing. For example, if the diameter of the network is $D(n)$, then the total communication cost to build a decentralized erasure code will be at most $O(D(n)n\ln k)$. To become more specific we need to impose additional assumptions that depend on the specific application. If $D(n) = O(\sqrt{n})$ for example in a grid network, the total communication cost would be bounded by $O(n^{1.5}\ln k)$ to make the data available in $k = O(n)$ storage nodes.

Since each data node is essentially multicasting its packet to $O(\ln k)$ storage nodes, multicast trees can be used to minimize the communication cost. These issues depend on the specific network model and geometry and we do not address them in this work.

2.4.1 Perimetric Storage

To perform some experimental evaluation and also to illustrate how the decentralized erasure codes can be used as a building block for more complex applications, we consider the following scenario. Suppose we have N total nodes placed on a grid in the unit square (dense scaling) and we are only interested in storing information in

the $4\sqrt{N}$ nodes on the perimeter of the square (see Figure 2.5). This is an interesting extension since in most cases the sensor network will be monitoring an environment and potential users interested in the data will have easier access to the perimeter of this environment. Therefore we will have $n = 4\sqrt{N}$ storage nodes and $k = \rho\sqrt{N}$ data nodes for some constant $\rho < 4$. The k data nodes can be placed in the grid randomly or by using some optimized sensor placement strategy [Ganesan *et al.*, 2004]. Notice that we only have $O(\sqrt{N})$ nodes measuring or storing. The rest are used as relays and perhaps it is more interesting to assume that the k data nodes are duty-cycled to elongate the lifetime of the network. Note that in a dense network scenario $\sqrt{N}$ can become sufficiently large to monitor the environment of interest. Again, we want to query any k nodes from the perimeter and be able to reconstruct the original k data packets w.h.p. The problem now is that the diameter of the network (assuming greedy geographic routing) is $O(\sqrt{N}) = O(n)$ as opposed to $\sqrt{n}$.

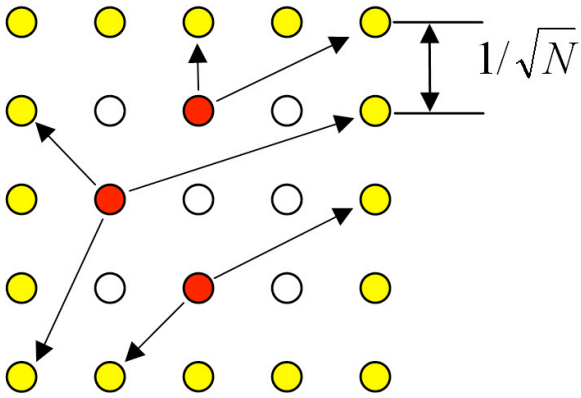


Figure 2.5: Perimetric storage: The $n = 4\sqrt{N}$ nodes on the perimeter are used as storage, and $k = O(\sqrt{N})$ nodes inside the grid are the data nodes.

We assume that the transmission radius is scaling like $O(\frac{1}{\sqrt{N}})$ and measure communication cost as the total number of 1-hop radio transmissions (each transfers one packet for one hop) required to build the decentralized erasure code. It can be easily seen that the total communication cost is at most $O(N \ln N)$ which yields a loga-

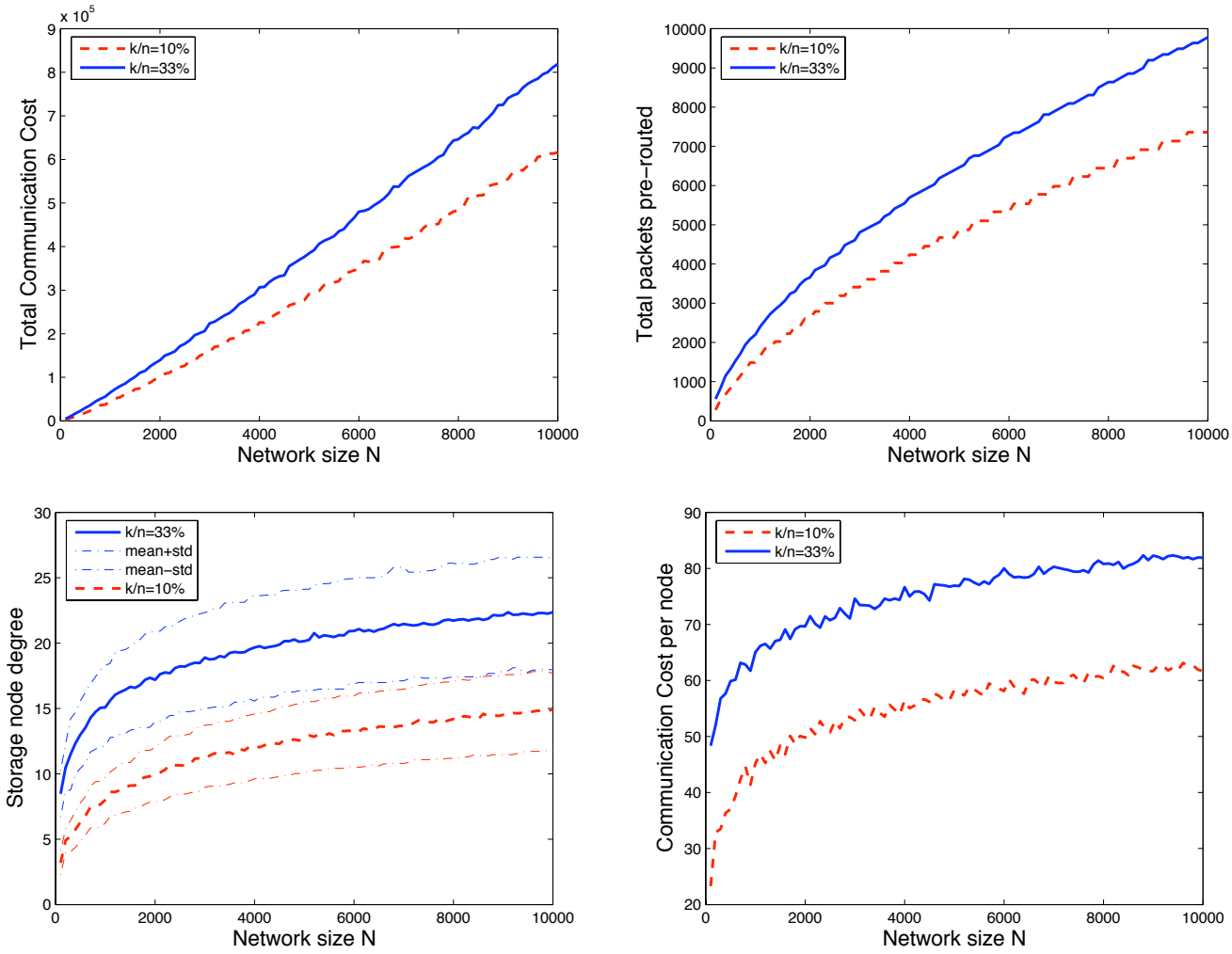


Figure 2.6: Experiments for perimetric storage scenario. For each subgraph, we plot $k/n = 10\%$ (so $k = 0.4\sqrt{N}$) and $k/n = 33\%$ (so $k = 4/3\sqrt{N}$). In both cases $n = 4\sqrt{N}$ a) Total communication cost to prepare the decentralized erasure code. b) Total number of packets pre-routed. c) Average and standard deviation plots for the number of packets that are stored at storage nodes. e) Total communication cost per node.

rithmic bound $O(\ln N)$ on the transmissions per node. Figure 2.6 illustrates some experiments on the performance under the perimetric storage scenario. Notice that the communication cost per node is indeed growing very slowly in N.

2.4.2 Correlated Data

For sensor network applications, the sensed data could be highly correlated and this correlation can be exploited to improve the performance [Slepian and Wolf, 1973]. Distributed Source Coding Using Syndromes (DISCUS) [Pradhan and Ramchandran, 2003] is a practical means of achieving this. The data nodes form the syndromes of the data packets they observe under suitable linear codes. These syndromes are treated as the data which the nodes pre-route to form the decentralized erasure codewords at the storage nodes. The data collector reconstructs the syndromes by gathering the packets from k storage nodes. Using DISCUS decoding the collector can recover the original data from the syndromes. The correlation statistics, which is required by DISCUS can be learned by observing previous data at the collection point. The data nodes only need to know the rates at which they will compress their packets. This can be either communicated to them or learned adaptively in a distributed network protocol. The syndromes can be considerably shorter than the original data packets if the data observed by the different nodes are significantly correlated as is usually the case in sensor networks. Note that this approach is separating the source coding problem from the storage problem and this may not be optimal in general as shown in [Ramamoorthy *et al.*, 2004].

Chapter 3

The Repair Problem

3.1 Introduction

In this chapter we define and investigate the problem of repairing failures of storage nodes with the minimal possible communication. Beyond sensor networks, the proposed repair schemes may have applications for data centers, RAID and peer-to-peer storage systems such as OceanStore [Rhea *et al.*, 2001], Total Recall [Bhagwan *et al.*, 2004], and DHash++ [Dabek *et al.*, 2004].

Consider a data object (or file) of total size $\mathcal{M}$, separated into k fragments, each of size $\mathcal{M}/k$. Consider using an (n, k) erasure coding scheme to generate n coded fragments which are subsequently stored in n distributed storage nodes. Then, the data object can be recovered from any set of k coded pieces. This performance is optimal in terms of the redundancy–reliability tradeoff because k pieces, each of size $\mathcal{M}/k$, provide the minimum data for recovering the file, which is of size $\mathcal{M}$. Several distributed file storage designs [Rhea *et al.*, 2003; Bhagwan *et al.*, 2004; Dabek *et al.*, 2004] use such erasure coding packets instead of replication.

However, a complication arises: In distributed storage systems, redundancy must

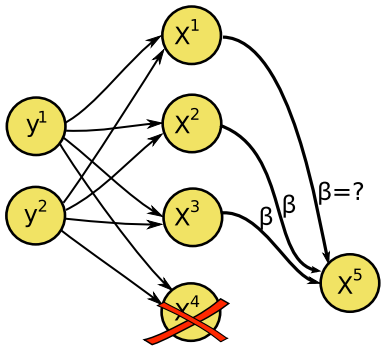


Figure 3.1: The repair problem: Assume that a (4,2) MDS erasure code is used to generate 4 fragments (stored in nodes $x^1, \ldots x^4$) with the property that any 2 can be used to reconstruct the original data y^1, y^2. When node x^4 fails, and a newcomer x^5 needs to generate an erasure fragment from $x^1, \ldots x^3$, what is the minimum amount of information that needs to be communicated?

be continually refreshed as nodes fail or leave the system, which involves large data transfers across the network. This problem is best illustrated in the simple example of Fig. 3.1: a data object is divided in two fragments y^1, y^2 (say, each of size 1Mb) and these encoded into four fragments $x^1, \ldots x^4$ of same size, with the property that any two out of the four can be used to recover the original y^1, y^2. Now assume that storage node x^4 fails and a new node x^5, the newcomer, needs to communicate with existing nodes and create a new encoded packet, such that any two out of x^1, x^2, x^3, x^5 suffice to recover the data object. Clearly, if the newcomer can download any two encoded fragments (say from x^1, x^2), reconstruction of the whole data object is possible and then a new encoded fragment can be generated (for example by making a new linear combination that is independent from the existing ones). This, however, requires the communication of 2Mb in the network to generate an erasure encoded fragment of size 1Mb at x^5. In general, if an object of size $\mathcal{M}$ is divided in k initial fragments, the repair bandwidth with this strategy is $\mathcal{M}$ bits to generate a fragment of size $\mathcal{M}/k$. In contrast, if replication is used instead, a new replica may simply be copied from any other existing node, incurring no bandwidth overhead. It was commonly believed

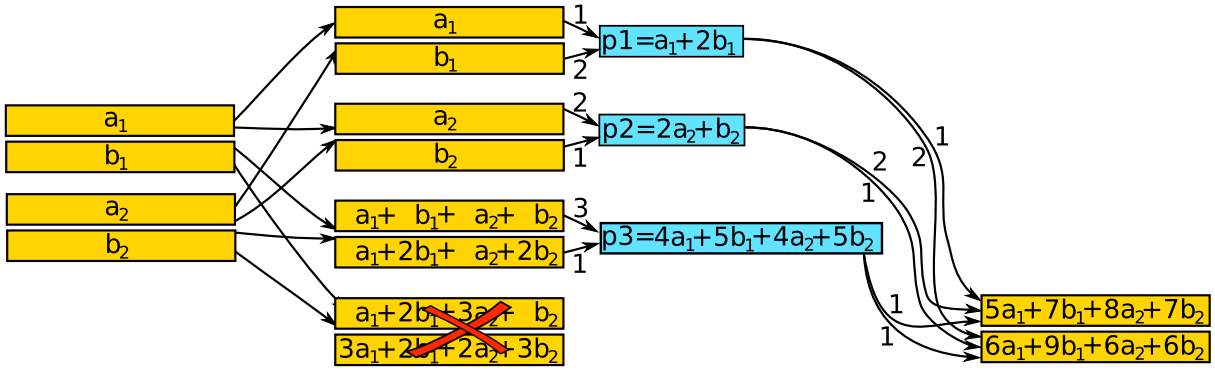


Figure 3.2: Example: A repair for a (4,2)-Minimum-Storage Regenerating Code. All the packets (boxes) in this figure have size 0.5Mb and each node stores two packets. Note that any two nodes have four equations that can be used to recover the data, a_1, a_2, b_1, b_2. The parity packets p_1, p_2, p_3 are used to create the two packets of the newcomer, requiring repair bandwidth of 1.5MB. The multiplying coefficients are selected at random and the example is shown over the integers for simplicity (although any sufficiently large field would be enough). The key point is that nodes do not send their information but generate smaller parity packets of their data, and forward them to the newcomer, who further mixes them to generate two new packets. Note that the selected coefficients also need to be included in the packets, which introduces some overhead.

that this k-factor overhead in repair bandwidth is an unavoidable overhead that comes with the benefits of coding (see, for example, [Rodrigues and Liskov, 2005]). Indeed, all known coding constructions require access to the original data object to generate encoded fragments.

In this chapter (based on [Dimakis *et al.*, 2007] and [Wu *et al.*, 2007]) we show that surprisingly, there exist erasure codes that can be repaired without communicating the whole data object. In particular, for the $(4, 2)$ example, we show that the newcomer can download 1.5Mb to repair a failure and that this is the information theoretic minimum (see Fig. 3.2 for an example). More generally, we identify a tradeoff between

storage and repair bandwidth and show that codes exist that achieve every point on this optimal tradeoff curve. We call codes that lie on this optimal tradeoff curve *regenerating codes*.

The two extremal points on the tradeoff curve are of special interest and we refer to them as minimum-storage regenerating (MSR) codes and minimum-bandwidth regenerating (MBR) codes. The former correspond to Maximum Distance Separable (MDS) codes that can also be efficiently repaired. At the other end of the tradeoff are the MBR codes, which have minimum repair bandwidth. We show that if each storage node is allowed to store slightly more than $\mathcal{M}/k$ bits, the repair bandwidth can be significantly reduced.

The remainder of this chapter is organized as follows. In Section 3.2 we discuss relevant background and related work from network coding theory and distributed storage systems. In Section 3.3 we introduce the notion of the information flow graph, which represents how information is communicated and stored in the network as nodes join and leave the system. In Section 3.3.2 we characterize the minimum storage and repair bandwidth and show that there is a tradeoff between these two quantities that can be expressed in terms of a maximum flow on this graph. We further show that for any finite information flow graph, there exists a regenerating code that can achieve any point on the minimum storage/ bandwidth feasible region we computed. Finally, in Section 3.4 we evaluate the performance of the proposed regenerating codes using traces of failures in real systems and compare to alternative schemes previously proposed in the distributed storage literature.

3.2 Background and Related Work

3.2.1 Erasure codes

Classical coding theory focuses on the tradeoff between redundancy and error tolerance. In terms of the redundancy-reliability tradeoff, the Maximum Distance Separable (MDS) codes are optimal. The most well-known class of MDS erasure codes is the Reed-Solomon code. More recent studies on erasure coding focus on other performance metrics. For instance, sparse graph codes [Luby *et al.*, 2001; Luby, 2002; Shokrollahi, 2006] can achieve near-optimal performance in terms of the redundancy-reliability tradeoff and also require low encoding and decoding complexity. Another line of research for erasure coding in storage applications is parity array codes; see, e.g., [Blaum *et al.*, 1995; Xu and Bruck, 1999; Huang and Xu, 2005; Hafner, 2005]. The array codes are based solely on XOR operations and they are generally designed with the objective of low encoding, decoding, and update complexities. Plank [Plank and Thomason, 2004] gave a tutorial on erasure codes for storage applications at USENIX FAST 2005, which covers Reed-Solomon codes, parity-array codes, and LDPC codes.

Compared to these studies, we focus on different performance metrics. Specifically, motivated by practical concerns in large distributed storage systems, we explore erasure codes that offer good tradeoffs in terms of redundancy, reliability, and repair bandwidth tradeoff.

3.2.2 Network Coding

Network coding is a generalization of the conventional routing (store-and-forwarding) method. In conventional routing, each intermediate node in the network simply stores and forwards information received. In contrast, network coding allows the

intermediate nodes to generate output data by encoding (i.e., computing certain functions of) previously received input data. Thus, network coding allows information to be "mixed" at intermediate nodes. The potential advantages of network coding over routing include resource (e.g., bandwidth and power) efficiency, computational efficiency, and robustness to network dynamics. As shown by the pioneering work of Ahlswede et al. [Ahlswede *et al.*, 2000b], network coding can increase the possible network throughput, and in the multicast case can achieve the maximum data rate theoretically possible.

Subsequent work [Li *et al.*, 2003; Koetter and Médard, 2003] showed that the maximum multicast capacity can be achieved by using linear encoding functions at each node. The studies by Ho *et al.* [Ho *et al.*, 2006b] and Sanders *et al.* [Sander *et al.*, 2003] further showed that random linear network coding over a sufficiently large finite field can (asymptotically) achieve the multicast capacity. A polynomial complexity procedure to construct deterministic network codes that achieve the multicast capacity is given by Jaggi *et al.* [Jaggi *et al.*, 2005].

For distributed storage, network coding was introduced in [Dimakis *et al.*, 2005; Dimakis *et al.*, 2006a] for wireless sensor networks. Many aspects of coding were explored [Kamra *et al.*, 2006; Huang *et al.*, 2007; Wang *et al.*, 2006; Jiang, 2006] for networked storage applications.

The key difference of this work to this existing literature is that we bring the dimension of *repair bandwidth* into the picture, and present fundamental bounds and constructions for network codes that need to be maintained over time. Similar to this related work, intermediate nodes form linear combinations in a finite field and the combination coefficients are also stored in each packet, creating some overhead that can be made arbitrarily small for larger packet sizes. In regenerating codes, repair bandwidth is reduced because many nodes create small parity packets of their data that essentially contain enough novel information to generate a new encoded

fragment, without requiring to reconstruct the whole data object.

3.2.3 Distributed storage systems

A number of recent studies [Rhea *et al.*, 2003; Dabek *et al.*, 2001; Rowstron and Druschel, 2001; Bhagwan *et al.*, 2004; Weatherspoon *et al.*, 2005] have designed and evaluated large-scale, peer-to-peer distributed storage systems. Redundancy management strategies for such systems have been evaluated in [Weatherspoon and Kubiatowicz, 2002b; Blake and Rodrigues, 2003; Bhagwan *et al.*, 2004; Rodrigues and Liskov, 2005; Weatherspoon *et al.*, 2005; Chun *et al.*, 2006; Tati and Voelker, 2006; Godfrey *et al.*, 2006].

Among these, [Weatherspoon and Kubiatowicz, 2002b; Bhagwan *et al.*, 2004; Rodrigues and Liskov, 2005] compared replication with erasure codes in the bandwidth-reliability tradeoff space. The analysis of Weatherspoon and Kubiatowicz [Weatherspoon and Kubiatowicz, 2002b] showed that erasure codes could reduce bandwidth use by an order of magnitude compared with replication. Bhagwan et al. [Bhagwan *et al.*, 2004] came to a similar conclusion in a simulation of the Total Recall storage system.

Rodrigues and Liskov [Rodrigues and Liskov, 2005] propose a solution to the repair problem that we call the *Hybrid strategy*: one special storage node maintains one full replica in addition to multiple erasure-coded fragments. The node storing the replica can produce new fragments and send them to newcomers, thus transferring just $\mathcal{M}/k$ bytes for a new fragment. However, maintaining an extra replica on one node dilutes the bandwidth-efficiency of erasure codes and complicates system design. For example, if the replica is lost, new fragments cannot be created until it is restored. The authors show that in high-churn environments (i.e., high rate of node joins/leaves), erasure codes provide a large storage benefits but the bandwidth cost

is too high to be practical for a P2P distributed storage system, using the Hybrid strategy. In low-churn environments, the reduction in bandwidth is negligible. In moderate-churn environments, there is some benefit, but this may be outweighed by the added architectural complexity that erasure codes introduce as discussed further in Section 3.4.5. These conclusions were based on an analytical model augmented with parameters estimated from traces of real systems. Compared with [Weatherspoon and Kubiatowicz, 2002b], [Rodrigues and Liskov, 2005] used a much smaller value of k (7 instead of 32) and the Hybrid strategy to address the code regeneration problem. In Section 3.4, we follow the evaluation methodology of [Rodrigues and Liskov, 2005] to measure the performance of the two redundancy maintenance schemes that we introduce.

3.3 Analysis

Our analysis is based on a particular graphical representation of a distributed storage system, which we refer to as an *information flow graph* $\mathcal{G}$. This graph describes how the information of the data object is communicated through the network, stored in nodes with limited memory, and reaches reconstruction points at the data collectors.

3.3.1 Information Flow Graph

The information flow graph is a directed acyclic graph consisting of three kinds of nodes: a single data source S, storage nodes $\mathsf{x}^i_{in}, \mathsf{x}^i_{out}$ and data collectors DC_i. The single node S corresponds to the source of the original data. Storage node i in the system is represented by a storage input node x^i_{in}, and a storage output node x^i_{out}; these two nodes are connected by a directed edge $\mathsf{x}^i_{in} \rightarrow \mathsf{x}^i_{out}$ with capacity equal to the amount of data stored at node i. See Figure 3.3 for an illustration. Note

that Jiang [Jiang, 2006] independently proposed a construction very similar to the information flow graph, but for optimizing a different objective.

Given the dynamic nature of the storage systems that we consider, the information flow graph also evolves in time. At any given time, each vertex in the graph is either *active* or *inactive*, depending on whether it is available in the network. At the initial time, only the source node S is active; it then contacts an initial set of storage nodes, and connects to their inputs (x_{in}) with directed edges of infinite capacity. From this point onwards, the original source node S becomes and remains inactive. At the next time step, the initially chosen storage nodes become now active; they represent a distributed erasure code, corresponding to the desired steady state of the system. If a new node j joins the system, it can only be connected with active nodes. If the newcomer j chooses to connect with active storage node i, then we add a directed edge from x^i_{out} to x^j_{in}, with capacity equal to the amount of data that the newcomer downloads from node i. Note that in general it is possible for nodes to download more data than they store, as in the example of the $(4, 2)$-erasure code. If a node leaves the system, it becomes inactive. Finally, a data collector DC is a node that corresponds to a request to reconstruct the data. Data collectors connect to subsets of active nodes through edges with infinite capacity.

An important notion associated with the information flow graph is that of minimum cuts: A cut in the graph $\mathcal{G}$ between the source S and a fixed data collector node DC is a subset C of edges such that, there is no path starting from S to DC that does not have one or more edges in C. The minimum cut is the cut between S and DC in which the total sum of the edge capactities is smallest.

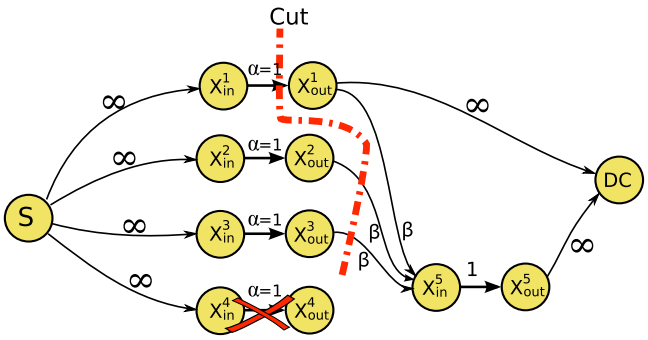


Figure 3.3: Illustration of the information flow graph $\mathcal{G}$ corresponding to the (4,2) code of figure 1. A distributed storage scheme uses an $(4, 2)$ erasure code in which any 2 fragments suffice to recover the original data. If node x^4 becomes unavailable and a new node joins the system, we need to construct new encoded fragment in x^5. To do so, node x^5_{in} is connected to the $d = 3$ active storage nodes. Assuming β bits communicated from each active storage node, of interest is the minimum β required. The min-cut separating the source and the data collector must be larger than $\mathcal{M} = 2$Mb for reconstruction to be possible. For this graph, the min-cut value is given by $1 + 2\beta$, implying that $\beta \geq 0.5$Mb is sufficient and necessary.

3.3.2 Storage-Bandwidth Tradeoff

We are now ready for the main result of this chapter, the characterization of the feasible storage-repair bandwidth points. The setup is as follows: The normal redundancy we want to maintain requires n active storage nodes, each storing α bits. Whenever a node fails, a newcomer downloads β bits each from any d surviving nodes. Therefore the total repair bandwidth is $\gamma = d\beta$ (see figure 3.3). We restrict our attention to the symmetric setup where it is required that any k storage nodes can recover the original file, and a newcomer downloads the same amount of information from each of the existing nodes.

For each set of parameters $(n, k, d, \alpha, \gamma)$, there is a family of information flow graphs, each of which corresponds to a particular evolution of node failures/repairs. We denote this family of directed acyclic graphs by $\mathcal{G}(n, k, d, \alpha, \gamma)$. An $(n, k, d, \alpha, \gamma)$ tuple will be feasible, if a code with storage α and repair bandwidth γ exists. For

the example in figure 3.3, the point $(4, 2, 3, 1\text{Mb}, 1.5\text{Mb})$ is feasible (and a code that achieves it is shown in figure 3.2) and also on the optimal tradeoff whereas a standard erasure code which communicates the whole data object would correspond to $\gamma = 2\text{Mb}$ instead. Note that n, k, d must be integers while α, β, γ are real valued.

Theorem 1. *For any $\alpha \geq \alpha^*(d, \gamma)$, the points $(n, k, d, \alpha, \gamma)$ are feasible, and linear network codes suffice to achieve them. It is information theoretically impossible to achieve points with $\alpha < \alpha^*(d, \gamma)$. The threshold function $\alpha^*(d, \gamma)$ (which also depends on n, k) is the following:*

$$\alpha^*(d, \gamma) = \begin{cases} \frac{\mathcal{M}}{k}, & \gamma \in [f(0), +\infty) \\ \frac{\mathcal{M} - g(i)\gamma}{k - i}, & \gamma \in [f(i), f(i-1)), \end{cases} \tag{3.1}$$

where

$$f(i) \triangleq \frac{2\mathcal{M}d}{(2k - i - 1)i + 2k(d - k + 1)}, \tag{3.2}$$

$$g(i) \triangleq \frac{(2d - 2k + i + 1)i}{2d}. \tag{3.3}$$

The minimum γ is

$$\gamma_{\text{min}} = f(k - 1) = \frac{2\mathcal{M}d}{2kd - k^2 + k}. \tag{3.4}$$

The complete proof of this theorem is given in the analysis section. The main idea is that the code repair problem can be mapped to a multicasting problem on the information flow graph. Known results on network coding for multicasting can then be used to establish that code repair can be achieved if and only if the underlying information flow graph has enough connectivity. The bulk of the technical analysis of the proof then involves computing the minimum cuts on arbitrary graphs

in $\mathcal{G}(n, k, d, \alpha, \gamma)$ and solving an optimization problem for minimizing α subject to a sufficient flow constraint.

The optimal tradeoff curves for $k = 5, n = 10, d = 9$ and $k = 10, n = 15, d = 14$ are shown in Figure 3.4 (top) and (bottom), respectively.

3.3.3 Special Cases: Minimum-Storage Regenerating (MSR) Codes and Minimum-Bandwidth Regenerating (MBR) Codes

We now study two extremal points on the optimal tradeoff curve, which correspond to the best storage efficiency and the minimum repair bandwidth, respectively. We call codes that attain these points minimum-storage regenerating (MSR) codes and minimum-bandwidth regenerating (MBR) codes, respectively.

It can be verified that the minimum storage point is achieved by the pair

$$(\alpha_{MSR}, \gamma_{MSR}) = \left(\frac{\mathcal{M}}{k}, \quad \frac{\mathcal{M}d}{k(d-k+1)} \right). \tag{3.5}$$

If we substitute $d = k$ into the above, we note that the total network bandwidth for repair is $\mathcal{M}$, the size of the original file. Therefore, if we only allow a newcomer to contact k nodes, it is optimal to download the whole file and then compute the new fragment. However, if we allow a newcomer to contact more than k nodes, the network bandwidth γ_{MSR} can be reduced significantly. The minimum network bandwidth is clearly achieved by having the newcomer contact all other nodes. For instance, for $(n, k) = (14, 7)$, the newcomer needs to download only $\frac{\mathcal{M}}{49}$ from each of the $d = n - 1 = 13$ active storage nodes, making the repair bandwidth equal to $\frac{13\mathcal{M}}{49}$, required to generate a fragment of size $\frac{\mathcal{M}}{7}$.

Since the MSR codes store $\frac{\mathcal{M}}{k}$ bits at each node while ensuring any k coded blocks can be used to recover the original file, the MSR codes have equivalent reliability-

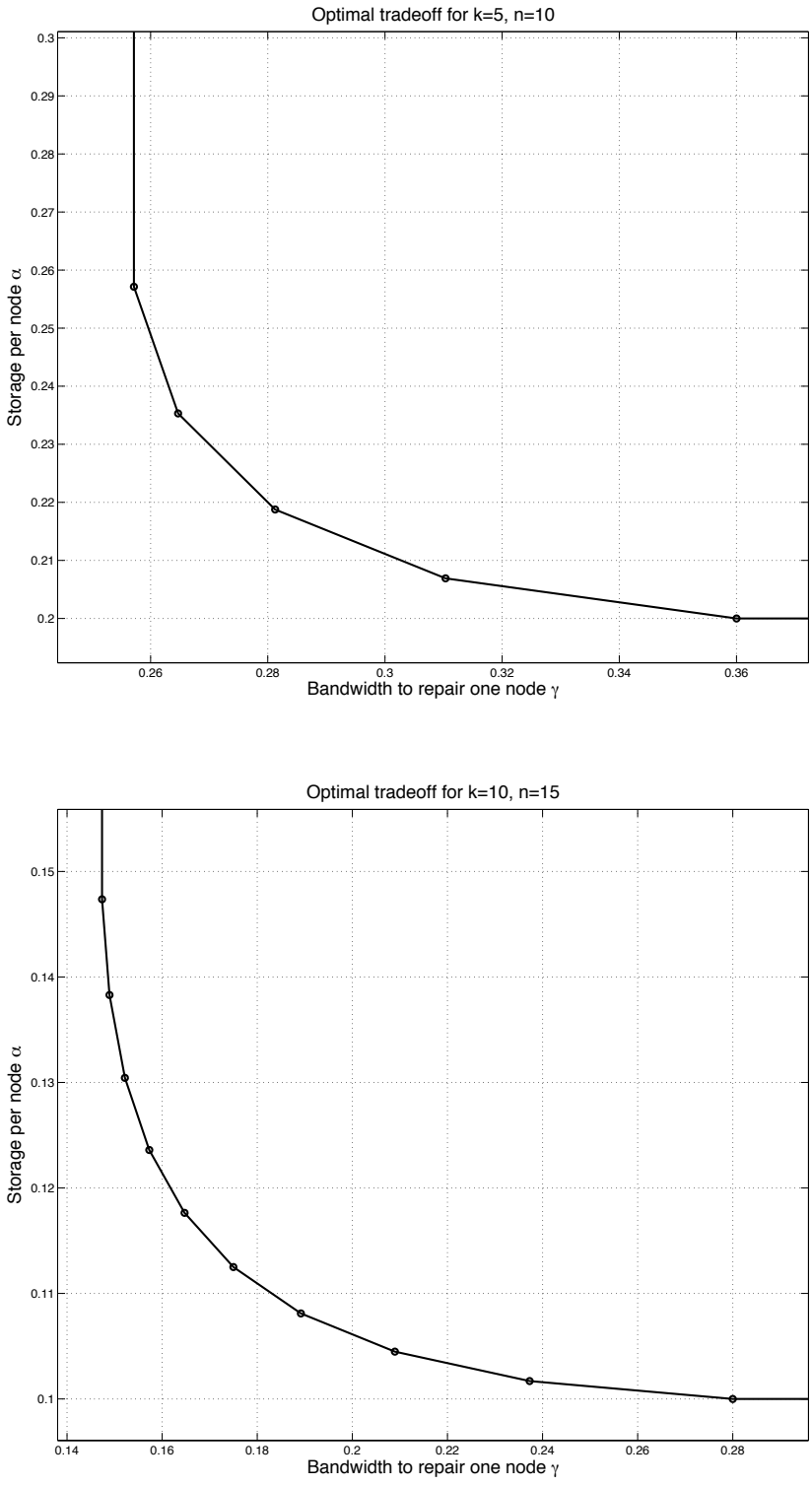


Figure 3.4: Optimal tradeoff curve between storage α and repair bandwidth γ, for $k = 5, n = 10$ (left) and $k = 10, n = 15$ (right). For both plots $\mathcal{M} = 1$ and $d = n - 1$. Note that traditional erasure coding corresponds to the points $(\gamma = 1, \alpha = 0.2)$ and $(\gamma = 1, \alpha = 0.1)$ for the top and bottom plots.

redundancy performance with standard Maximum Distance Separable (MDS) codes. However, MSR codes outperform classical MDS codes in terms of the network repair bandwidth.

At the other end of the tradeoff are MBR codes, which have minimum repair bandwidth. It can be verified that the minimum repair bandwidth point is achieved by

$$(\alpha_{MBR}, \gamma_{MBR}) = \left(\frac{2\mathcal{M}d}{2kd - k^2 + k}, \quad \frac{2\mathcal{M}d}{2kd - k^2 + k} \right). \tag{3.6}$$

Note that the minimum bandwidth regenerating codes, the storage size α is equal to γ, the total number of bits downloaded. Therefore MBR codes incur no bandwidth expansion at all, just like a replication system does. However, the benefit of MBR codes is significantly better storage efficiency.

3.4 Evaluation

In this section, we compare regenerating codes with other redundancy management schemes in the context of distributed storage systems. We follow the evaluation methodology of [Rodrigues and Liskov, 2005], which consists of a simple analytical model whose parameters are obtained from traces of node availability measured in several real distributed systems.

We begin in Section 3.4.1 with a discussion of node dynamics and the objectives relevant to distributed storage systems, namely reliability, bandwidth, and disk space. We introduce the model in Section 3.4.2 and estimate realistic values for its parameters in Section 3.4.3. Section 3.4.4 contains the quantitative results of our evaluation. In Section 3.4.5, we discuss qualitative tradeoffs between regenerating codes and other strategies, and how our results change the conclusion of [Rodrigues and Liskov, 2005]

that erasure codes provide limited practical benefit.

3.4.1 Node dynamics and objectives

In this section we introduce some background and terminology which is common to most of the work discussed in Section 3.2.3.

We draw a distinction between *permanent* and *transient* node failures. A permanent failure, such as the permanent departure of a node from the system or a disk failure, results in loss of the data stored on the node. In contrast, data is preserved across a transient failure, such as a reboot or temporary network disconnection. We say that a node is *available* when its data can be retrieved across the network.

Distributed storage systems attempt to provide two types of reliability: availability and durability. A file is *available* when it can be reconstructed from the data stored on currently available nodes. A file's *durability* is maintained if it has not been lost due to permanent node failures: that is, it may be available at some point in the future. Both properties are desirable, but here we report results for availability only. Specifically, we will show *file unavailability*, the fraction of time that the file is not available.

3.4.2 Model

We use a model which is intended to capture the average-case bandwidth used to maintain a file in the system, and the resulting average availability of the file. With minor exceptions,[1]this model and the subsequent estimation of its parameters are equivalent to that of [Rodrigues and Liskov, 2005]. Although this evaluation method-

[1]In addition to evaluating a larger set of strategies and using a somewhat different set of traces, we count bandwidth cost due to permanent node failure only, rather than both failures and joins. Most designs [Bhagwan *et al.*, 2004; Weatherspoon *et al.*, 2005; Chun *et al.*, 2006] can avoid reacting to node joins. Additionally, we compute probabilities directly rather than using approximations to the binomial.

ology is a significant simplification of real storage systems, it allows us to compare directly with the conclusions of [Rodrigues and Liskov, 2005] as well as to calculate precise values for rare events.

The model has two key parameters, f and a. First, we assume that in expectation a fraction f of the nodes storing file data fail permanently per unit time, causing data transfers to repair the lost redundancy. Second, we assume that at any given time while a node is storing data, the node is available with some probability a (and with probability $1 - a$ is currently experiencing a transient failure). Moreover, the model assumes that the event that a node is available is independent of the availability of all other nodes.

Under these assumptions, we can compute the expected availability and maintenance bandwidth of various redundancy schemes to maintain a file of $\mathcal{M}$ bytes. We make use of the fact that for all schemes except MSR codes, the amount of bandwidth used is equal to the amount of redundancy that had to be replaced, which is in expectation f times the amount of storage used.

Replication: If we store $\mathcal{R}$ replicas of the file, then we store a total of $\mathcal{R} \cdot \mathcal{M}$ bytes, and in expectation we must replace $f \cdot \mathcal{R} \cdot \mathcal{M}$ bytes per unit time. The file is unavailable if no replica is available, which happens with probability $(1 - a)^{\mathcal{R}}$.

Ideal Erasure Codes: For comparison, we show the bandwidth and availability of a hypothetical (n, k) erasure code strategy which can "magically" create a new packet while transferring just $\mathcal{M}/k$ bytes (*i.e.*, the size of the packet). Setting $n = k \cdot \mathcal{R}$, this strategy sends $f \cdot \mathcal{R} \cdot \mathcal{M}$ bytes per unit time and has unavailability probability $U_{\text{ideal}}(n, k) := \sum_{i=0}^{k-1} \binom{n}{i} a^i (1 - a)^{n-i}$.

Hybrid: If we store one full replica plus an (n, k) erasure code where $n = k \cdot (\mathcal{R} - 1)$, then we again store $\mathcal{R} \cdot \mathcal{M}$ bytes in total, so we transfer $f \cdot \mathcal{R} \cdot \mathcal{M}$ bytes per unit time in expectation. The file is unavailable if the replica is unavailable *and*

fewer than k erasure-coded packets are available, which happens with probability $(1-a)\cdot U_{\text{ideal}}(n,k)$.

Minimum-Storage Regenerating Codes: An (n,k) MSR Code with redundancy $\mathcal{R}=n/k$ stores $\mathcal{RM}$ bytes in total, so $f\cdot\mathcal{R}\cdot\mathcal{M}$ bytes must be replaced per unit time. We will refer to the *overhead* of an MSR code δ_{MSR} as the extra amount of information that needs to be transfered compared to the fragment size $\mathcal{M}/k$:

$$\delta_{MSR} \triangleq \frac{(n-1)\beta_{MSR}}{\mathcal{M}/k} = \frac{n-1}{n-k}. \tag{3.7}$$

Therefore, replacing a fragment requires transferring over the network δ_{MSR} times the size of the fragment in the most favorable case when newcomers connect to $d=n-1$ nodes to construct a new fragment. Therefore, this results in $f\cdot\mathcal{R}\cdot\mathcal{M}\cdot\delta_{\text{MSR}}$ bytes sent per unit time, and unavailability $U_{\text{ideal}}(n,k)$.

Minimum-Bandwidth Regenerating Codes:

It is convenient to define the MBR code overhead as the amount of information transfered over the ideal fragment size:

$$\delta_{MBR} \triangleq \frac{(n-1)\beta_{MBR}}{\mathcal{M}/k} = \frac{2(n-1)}{2n-k-1}. \tag{3.8}$$

Therefore, an (n,k) MBR Code stores $\mathcal{M}\cdot n\cdot\delta_{\text{MBR}}$ bytes in total. So in expectation $f\cdot\mathcal{M}\cdot n\cdot\delta_{\text{MBR}}$ bytes are transfered per unit time, and the unavailability is again $U_{\text{ideal}}(n,k)$.

3.4.3 Estimating f and a

In this section we describe how we estimate f, the fraction of nodes that permanently fail per unit time, and a, the mean node availability, based on traces of node availability in several distributed systems.

Trace	**Length (days)**	**Start date**	**Mean # nodes up**	f **(fraction failed per day)**	a
PlanetLab	527	Jan. 2004	303	0.017	0.97
Microsoft PCs	35	Jul. 6, 1999	41970	0.038	0.91
Skype	25	Sept. 12, 2005	710	0.12	0.65
Gnutella	2.5	May, 2001	1846	0.30	0.38

Table 3.1: The availability traces used in this study.

We use four traces of node availability with widely varying characteristics, summarized in Table 3.1. The **PlanetLab All Pairs Ping** [**Stribling, **] trace is based on pings sent every 15 minutes between all pairs of 200-400 nodes in PlanetLab, a stable, managed network research testbed. We consider a node to be up in one 15-minute interval when at least half of the pings sent to it in that interval succeeded. In a number of periods, all or nearly all PlanetLab nodes were down, most likely due to planned system upgrades or measurement errors. To exclude these cases, we "cleaned" the trace as follows: for each period of downtime at a particular node, we remove that period (i.e. we consider the node up during that interval) when the average number of nodes up during that period is less than half the average number of nodes up over all time. The **Microsoft PCs** [**Bolosky *et al.*, 2000**] trace is derived from hourly pings to desktop PCs within Microsoft Corporation. The **Skype superpeers** [**Guha *et al.*, 2006**] trace is based on application-level pings at 30-minute intervals to nodes in the Skype superpeer network, which may approximate the behavior of a set of well-provisioned endhosts, since superpeers may be selected in part based on bandwidth availability [Guha *et al.*, 2006]. Finally, the trace of **Gnutella peers** [**Saroiu *et al.*, 2002**] is based on application-level pings to ordinary Gnutella peers at 7-minute intervals.

We next describe how we derive f and a from these traces. It is of key importance for the storage system to distinguish between permanent and transient failures

(defined in Section 3.4.1), since only the former requires bandwidth-intensive replacement of lost redundancy. Most systems use a *timeout* heuristic: when a node has not responded to network-level probes after some period of time t, it is considered to have failed permanently. To approximate a storage system's behavior, we use the same heuristic. Node availability a is then calculated as the mean (over time) fraction of nodes which were available among those which were not considered permanently failed at that time.

The resulting values of f and a appear in Table 3.1, where we have fixed the timeout t at 1 day. Longer timeouts reduce overall bandwidth costs [Rodrigues and Liskov, 2005; Chun *et al.*, 2006], but begin to impact durability [Chun *et al.*, 2006] and are more likely to produce artificial effects in the short (2.5-day) Gnutella trace.

We emphasize that the procedure described above only provides an estimate of f and a which may be biased in several ways. Some designs [Chun *et al.*, 2006] reincorporate data on nodes which return after transient failures which were longer than the timeout t, which would reduce f. Additionally, even placing files on uniform-random nodes results in selecting nodes that are more available [Tati and Voelker, 2006] and less prone to failure [Godfrey *et al.*, 2006] than the average node. Finally, we have not accounted for the time needed to transfer data onto a node, during which it is effectively unavailable. However, we consider it unlikely that these biases would impact our main results since we are primarily concerned with the *relative* performance of the strategies we compare.

3.4.4 Quantitative results

Figure 3.5 shows the tradeoff between mean unavailability and mean maintenance bandwidth in each of the strategies of Section 3.4.2 using the values of f and a from Section 3.4.3 and $k = 7$. Feasible points in the tradeoff space are produced by varying

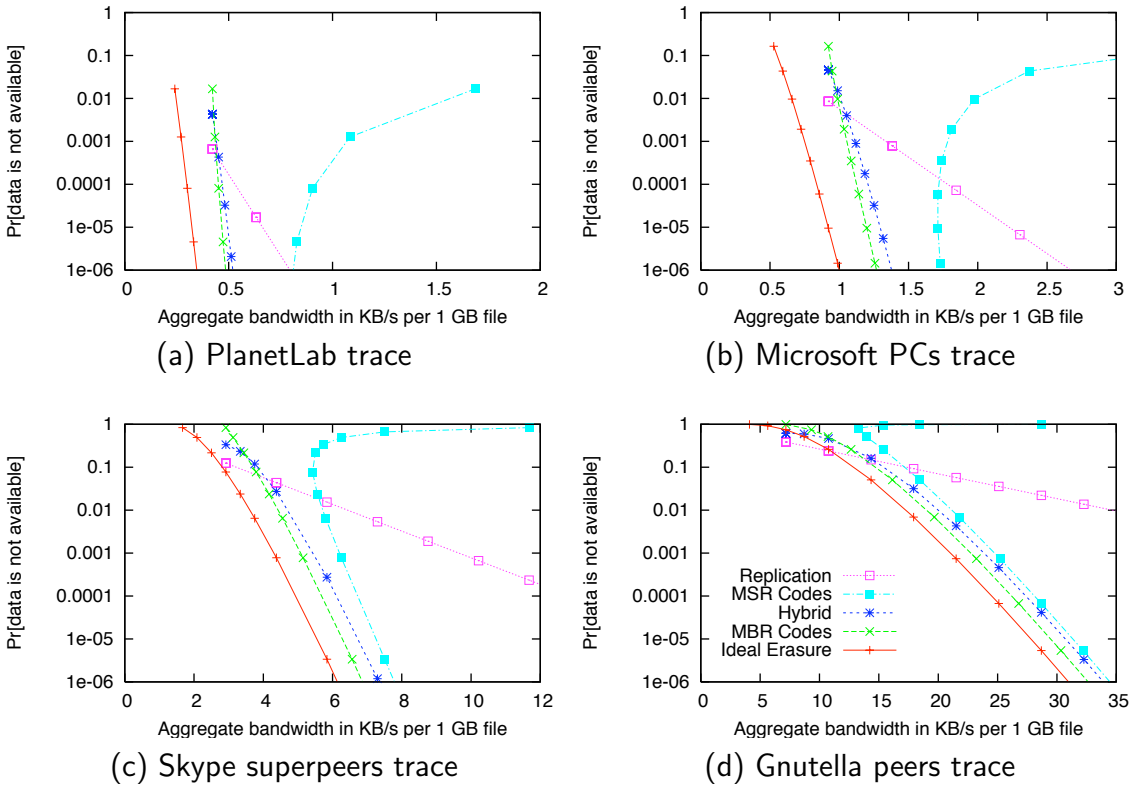
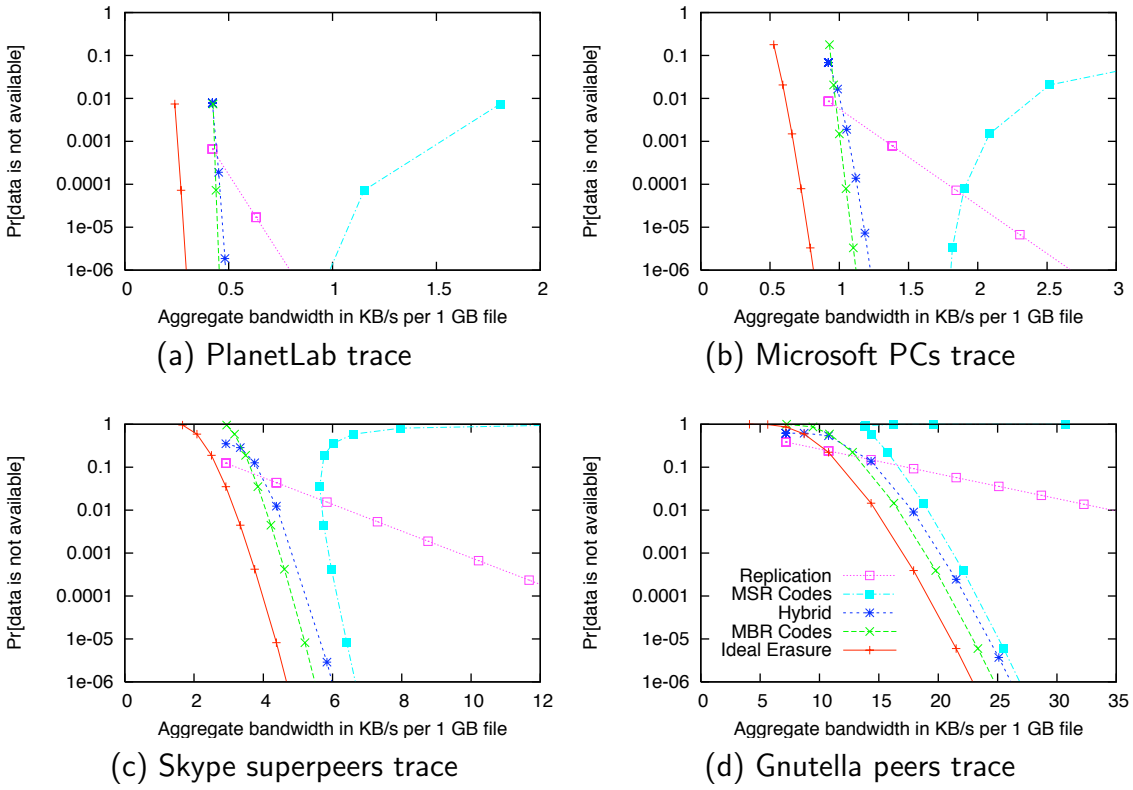


Figure 3.5: Availability-bandwidth tradeoff for $k = 7$ with parameters derived from each of the traces. The key in (d) applies to all four plots.



(a) PlanetLab trace

(b) Microsoft PCs trace

(c) Skype superpeers trace

(d) Gnutella peers trace

Figure 3.6: Availability-bandwidth tradeoff for $k = 14$ with parameters derived from each of the traces.

the redundancy factor $\mathcal{R}$. The marked points along each curve highlight a subset of the feasible points (i.e., points for which n is integral).

Figure 3.6 shows that relative performance of the various strategies is similar for $k = 14$.

For conciseness, we omit plots of storage used by the schemes. However, disk usage is proportional to bandwidth for all schemes we evaluate in this section, with the exception of minimum storage regenerating codes. This is because MSR codes are the only scheme in which the data transferred onto a newcomer is not equal to the amount of data that the newcomer finally stores. Instead, the storage used by MSR codes is equal to that of the storage used by hypothetical ideal erasure codes, and hence MSR codes' space usage is proportional to the bandwidth used by ideal codes.

For example, from Figure 3.5(b) we can compare the strategies at their feasible points closest to unavailability 0.0001, i.e., four nines of availability. At these points, MSR codes use about 44% more bandwidth and 28% less storage space than Hybrid, while MBR codes use about 3.7% less bandwidth and storage space than Hybrid. Additionally, these feasible points give MSR and MBR codes somewhat better unavailability than Hybrid (.000059 vs. 0.00018).

One interesting effect apparent in the plots is that MSR codes' maintenance bandwidth actually *decreases* as the redundancy factor $\mathcal{R}$ increases, before coming to a minimum and then increasing again. Intuitively, while increasing $\mathcal{R}$ increases the total amount of data that needs to be maintained, for small $\mathcal{R}$ this is more than compensated for by the reduction in overhead. The expected maintenance bandwidth per unit time is

$$f\,\mathcal{M}\,\mathcal{R}\,\delta_{\text{MSR}} = f\,\mathcal{M}\frac{n}{k}\frac{n-1}{n-k}. \tag{3.9}$$

It is easy to see that this function is minimized by selecting n one of the two integers

closest to

$$n_{opt} = k + \sqrt{k^2 - k}. \tag{3.10}$$

which approaches a redundancy factor of 2 as $k \to \infty$.

3.4.5 Qualitative comparison

In this section we discuss two questions: First, based on the results of the previous section, what are the qualitative advantages and disadvantages of the two extremal regenerating codes compared with the Hybrid coding scheme? Second, do our results affect the conclusion of Rodrigues and Liskov [Rodrigues and Liskov, 2005] that erasure codes offer too little improvement in bandwidth use to clearly offset the added complexity that they add to the system?

3.4.5.1 Comparison with Hybrid

Compared with Hybrid, for a given target availability, minimum storage regenerating codes offer slightly lower maintenance bandwidth and storage, and a simpler system architecture since only one type of redundancy needs to be maintained. An important practical disadvantage of using the Hybrid scheme is asymmetric design which can cause the disk I/O to become the bottleneck of the system during repairs. This is because the disc storing the full replica and generates the encoded fragments need to read the whole data object and compute the encoded fragment.

However, MBR codes have at least two disadvantages. First, constructing a new packet, or reconstructing the entire file, requires communcation with $n - 1$ nodes[2] rather than one (in Hybrid, the node holding the single replica). This adds overhead that could be significant for sufficiently small files or sufficiently large n. Perhaps

[2]The scheme could be adapted to connect to fewer than $n - 1$ nodes, but this would increase maintenance bandwidth.

more importantly, there is a factor δ_{MBR} increase in total data transferred to *read* the file, roughly 30% for a redundancy factor $\mathcal{R} = 2$ and $k = 7$ or 13% for $\mathcal{R} = 4$, Thus, if the frequency that a file is read is sufficiently high and k is sufficiently small, this inefficiency could become unacceptable. Again compared with Hybrid, MSR codes offer a simpler, symmetric system design and somewhat lower storage space for the same reliability. However, MSR codes have somewhat higher maintenance bandwidth and like MSB codes require that newcomers and data collectors connect to multiple nodes.

Rodrigues et al. [Rodrigues and Liskov, 2005] discussed two principal disadvantages of using erasure codes in a widely distributed system: coding—in particular, the Hybrid strategy—complicates the system architecture; and the improvement in maintenance bandwidth was minimal in more stable environments, which are the more likely deployment scenario. Regenerating codes address the first of these issues, which may make coding more broadly applicable.

3.5 Analysis and Proofs

Here we prove Theorem 1. We start with the following simple lemma.

Lemma 1. *No data collector* DC *can reconstruct the initial data object if the minimum cut in* $\mathcal{G}$ *between* S *and* DC *is smaller than the initial object size* $\mathcal{M}$*.*

Proof. The information of the initial data object must be communicated from the source to the particular data collector. Since every link in the information flow graph can only be used at most once, and since the point-to-point capacity is less than the data object size, a standard cut-set bound shows that the entropy of the data object conditioned on everything observable to the data collector is non-zero and therefore reconstruction is impossible. ■

The information flow graph casts the original storage problem as a network communication problem where the source s multicasts the file to the set of all possible data collectors. By analyzing the connectivity in the information flow graph, we obtain necessary conditions for all possible storage codes, as shown in Lemma 1. In addition to providing necessary conditions for all codes, the information flow graph can also imply the existence of codes under proper assumptions.

Proposition 1. *Consider any given finite information flow graph $\mathcal{G}$, with a finite set of data collectors. If the minimum of the min-cuts separating the source with each data collector is larger or equal to the data object size $\mathcal{M}$, then there exists a linear network code defined over a sufficiently large finite field $\mathbb{F}$ (whose size depends on the graph size) such that all data collectors can recover the data object. Further, randomized network coding guarantees that all collectors can recover the data object with probability that can be driven arbitrarily high by increasing the field size.*

Proof. The key point is observing that the reconstruction problem reduces exactly to multicasting on all the possible data collectors on the information flow graph $\mathcal{G}$. Therefore, the result follows directly from the constructive results in network coding theory for single source multicasting; see the discussion of related works on network coding in Section 3.2.2. ■

To apply Proposition 1, consider an information flow graph $\mathcal{G}$ that enumerates all possible failure/repair patterns and all possible data collectors when the number of failures/repairs is bounded. This implies that there exists a valid regenerating code achieving the necessary cut bound (cf. Lemma 1), which can tolerate a bounded number of failures/repairs. In another paper [Wu *et al.*, 2007], we present coding methods that construct deterministic regenerating codes that can tolerate infinite number of failures/repairs, with a bounded field size, assuming only the population

of active nodes at any time is bounded. For the detailed coding theoretic construction, refer to [Wu *et al.*, 2007].

We analyze the connectivity in the information flow graph to find the minimum repair bandwidth. The next key lemma characterizes the flow in any information flow graph, under arbitrary failure pattern and connectivity.

Lemma 2. *Consider any (potentially infinite) information flow graph G, formed by having n initial nodes that connect directly to the source and obtain α bits, while additional nodes join the graph by connecting to d existing nodes and obtaining β bits from each.*[3] *Any data collector t that connects to a k-subset of "out-nodes" (c.f. Figure 3.3) of G must satisfy:*

$$mincut(s,t) \geq \sum_{i=0}^{\min\{d,k\}-1} \min\{(d-i)\beta, \alpha\}. \tag{3.11}$$

Furthermore, there exists an information flow graph $G^* \in \mathcal{G}(n,k,d,\alpha,\beta)$ where this bound is matched with equality.

Proof: First, we show that there exists an information flow graph G^* where the bound (3.11) is matched with equality. This graph is illustrated by Figure 3.7. In this graph, there are initially n nodes labeled from 1 to n. Consider k newcomers labeled as $n+1,\ldots,n+k$. The newcomer node $n+i$ connects to nodes $n+i-d,\ldots,n+i-1$. Consider a data collector t that connects to the last k nodes, i.e., nodes $n+1,\ldots,n+k$. Consider a cut $(U,\overline{U})$ defined as follows. For each $i \in \{1,\ldots,k\}$, if $\alpha \leq (d-i)\beta$, then we include x_{out}^{n+i} in $\overline{U}$; otherwise, we include x_{out}^{n+i} and x_{in}^{n+i} in $\overline{U}$. Then this cut $(U,\overline{U})$ achieves (3.11) with equality.

[3]Note that this setup allows more graphs than those in $\mathcal{G}(n,k,d,\alpha,\beta)$. In a graph in $\mathcal{G}(n,k,d,\alpha,\beta)$, at any time there are n active storage nodes and a newcomer can only connect to the active nodes. In contrast, in a graph G described in this lemma, there is no notion of "active nodes" and a newcomer can connect to any d existing nodes.

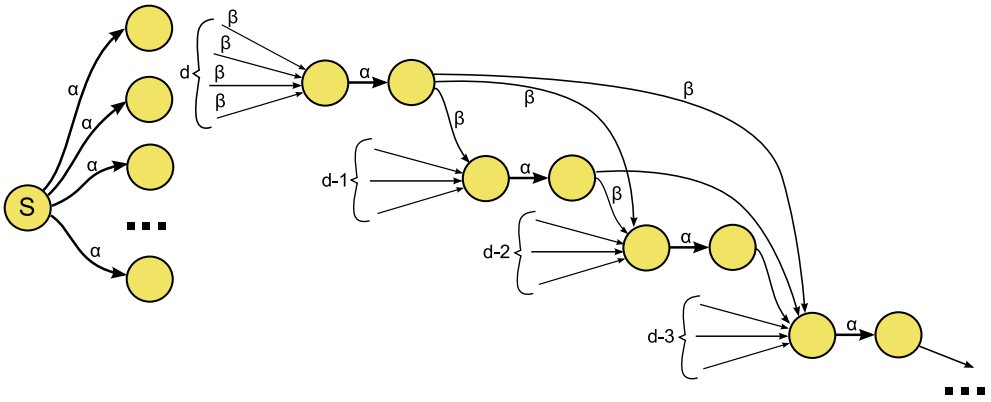


Figure 3.7: G^* used in the proof of lemma 2

We now show that (3.11) must be satisfied for any G formed by adding d in-degree nodes as described above. Consider a data collector t that connects to a k-subset of "out-nodes", say $\{\mathsf{x}^i_{out} : i \in I\}$. We want to show that any s–t cut in G has capacity at least

$$\sum_{i=0}^{\min\{d,k\}-1} \min\{(d-i)\beta, \alpha\}. \tag{3.12}$$

Since the incoming edges of t all have infinite capacity, we only need to examine the cuts $(U, \overline{U})$ with $s \in U$,

$$\mathsf{x}^i_{out} \in \overline{U}, \forall i \in I. \tag{3.13}$$

Let $\mathcal{C}$ denote the edges in the cut, i.e., the set of edges going from U to $\overline{U}$.

Every directed acyclic graph has a topological sorting (see, e.g., [Bang-Jensen and Gutin, 2001]), where a topological sorting (or acyclic ordering) is an ordering of its vertices such that the existence of an edge from v_i to v_j implies $i < j$. Let x^1_{out} be the topologically first output node in $\overline{U}$. Consider two cases:

- If $\mathsf{x}^1_{in} \in U$, then the edge $\mathsf{x}^1_{in}\mathsf{x}^1_{out}$ must be in $\mathcal{C}$.
- If $\mathsf{x}^1_{in} \in \overline{U}$, since x^1_{in} has an in-degree of d and it is the topologically first node in $\overline{U}$, all the incoming edges of x^1_{in} must be in $\mathcal{C}$.

Therefore, these edges related to x^1_{out} will contribute a value of $\min\{d\beta, \alpha\}$ to the cut capacity.

Now consider x^2_{out}, the topologically second output node in $\overline{U}$. Similar to the above, we have two cases:

- If $\mathsf{x}^2_{in} \in U$, then the edge $\mathsf{x}^2_{in}\mathsf{x}^2_{out}$ must be in $\mathcal{C}$.
- If $\mathsf{x}^2_{in} \in \overline{U}$, since at most one of the incoming edges of x^2_{in} can be from x^1_{out}, $d-1$ incoming edges of x^1_{in} must be in $\mathcal{C}$.

Following the same reasoning we find that for the i-th node ($i = 0, \ldots, \min\{d, k\} - 1$) in the sorted set $\overline{U}$, either one edge of capacity α or $(d-i)$ edges of capacity β must be in $\mathcal{C}$. Equation (3.11) is exactly summing these contributions. ∎

From Lemma 2, we know that there exists a graph $G^* \in \mathcal{G}(n, k, d, \alpha, \beta)$ whose mincut is exactly $\sum_{i=0}^{\min\{d,k\}-1} \min\{(d-i)\beta, \alpha\}$. This implies that if we want to ensure recoverability while allowing a newcomer to connect to *any* set of d existing nodes, then the following is a necessary condition[4]

$$\sum_{i=0}^{\min\{d,k\}-1} \min\{(d-i)\beta, \alpha\} \geq \mathcal{M}. \tag{3.14}$$

Furthermore, when this condition is satisfied, we know any graph in $\mathcal{G}(n, k, d, \alpha, \beta)$ will have enough flow from the source to each data collector. For this reason, we say

$$\mathsf{C} \triangleq \sum_{i=0}^{\min\{d,k\}-1} \min\{(d-i)\beta, \alpha\} \tag{3.15}$$

is the *capacity* for (n, k, d, α, β) regenerating codes (where each newcomer can access any arbitrary set of k nodes).

[4]This, however, does not rule out the possibility that the mincut is larger if a newcomer can choose the d existing nodes to connect to. We leave this as a future work.

Note that if $d < k$, requiring any d storage nodes to have a flow of $\mathcal{M}$ will lead to the same condition (c.f. (3.14)) as requiring any k storage nodes to have a flow of $\mathcal{M}$. Hence in such a case, we might as well set k as d. For this reason, in the following we assume $d \geq k$ without loss of generality.

We are interested in characterizing the achievable tradeoffs between the storage α and the repair bandwidth $d\beta$. To derive the optimal tradeoffs, we can fix the repair bandwidth and solve for the minimum α such that (3.14) is satisfied. Recall that $\gamma = d\beta$ the total repair bandwidth, and the parameters $(n, k, d, \alpha, \gamma)$ can be used to characterize the system. We are interested in finding the whole region of feasible points (α, γ) and then select the one that minimizes storage α or repair bandwidth γ. Consider fixing both γ and d (to some integer value) and minimize α;

$$\begin{aligned} \alpha^*(d,\gamma) \triangleq \min \quad & \alpha \\ \text{subject to:} \quad & \sum_{i=0}^{k-1} \min\left\{\left(1-\frac{i}{d}\right)\gamma, \alpha\right\} \geq \mathcal{M}. \end{aligned} \tag{3.16}$$

Now observe that the dependence on d must be monotone:

$$\alpha^*(d+1,\gamma) \leq \alpha^*(d,\gamma). \tag{3.17}$$

This is because $\alpha^*(d,\gamma)$ is always a feasible solution for the optimization for $\alpha^*(d+1,\gamma)$. Hence a larger d always implies a better storage–repair bandwidth tradeoff.

The optimization (3.16) can be explicitly solved: We call the solution, the threshold function $\alpha^*(d,\gamma)$, which for a fixed d, is piecewise linear:

$$\alpha^*(d,\gamma) = \begin{cases} \frac{\mathcal{M}}{k}, & \gamma \in [f(0), +\infty) \\ \frac{\mathcal{M}-g(i)\gamma}{k-i}, & \gamma \in [f(i), f(i-1)), \end{cases} \tag{3.18}$$

where

$$f(i) \triangleq \frac{2\mathcal{M}d}{(2k-i-1)i+2k(d-k+1)}, \tag{3.19}$$

$$g(i) \triangleq \frac{(2d-2k+i+1)i}{2d}. \tag{3.20}$$

The last part of the proof involves showing that the threshold function is the solution of this optimization. To simplify notation, introduce

$$b_i \triangleq \left(1-\frac{k-1-i}{d}\right)\gamma, \quad \text{for } i=0,\ldots,k-1. \tag{3.21}$$

Then the problem is to minimize α subject to the constraint:

$$\sum_{i=0}^{k-1}\min\{b_i,\alpha\} \geq B. \tag{3.22}$$

The left hand side of (3.22), as a function of α, is a piecewise-linear function of α:

$$\mathsf{C}(\alpha) = \begin{cases} k\alpha, & \alpha \in [0,b_0] \\ b_0+(k-1)\alpha, & \alpha \in (b_0,b_1] \\ \vdots & \vdots \\ b_0+\ldots+b_{k-2}+\alpha, & \alpha \in (b_{k-2},b_{k-1}] \\ b_0+\ldots+b_{k-1}, & \alpha \in (b_{k-1},\infty) \end{cases}. \tag{3.23}$$

Note from this expression that $\mathsf{C}(\alpha)$ is strictly increasing from 0 to its maximum value $b_0+\ldots+b_{k-1}$ as α increases from 0 to b_{k-1}. To find the minimum α such that

$\mathsf{C}(\alpha) \geq B$, we simply let $\alpha^* = \mathsf{C}^{-1}(B)$ if $B \leq b_0 + \ldots + b_{k-1}$:

$$\alpha^* = \begin{cases} \frac{B}{k}, & B \in [0, kb_0] \\ \frac{B-b_0}{k-1}, & B \in (kb_0, b_0 + (k-1)b_1] \\ \vdots & \vdots \\ B - \sum_{j=0}^{k-2} b_j, & B \in \left(\sum_{j=0}^{k-2} b_j + b_{k-2},\ \sum_{j=0}^{k-1} b_j\right] \end{cases} \tag{3.24}$$

For $i = 1, \ldots, k-1$, the i-th condition in the above expression is:

$$\alpha* = \frac{B - \sum_{j=0}^{i-1} b_j}{k-i},$$
$$\text{for } B \in \left(\sum_{j=0}^{i-1} b_j + (k-i)b_{i-1}, \sum_{j=0}^{i} b_j + (k-i-1)b_i\right],$$

Note from the definition of $\{b_i\}$ (3.21) that

$$\begin{aligned} \sum_{j=0}^{i-1} b_j &= \sum_{j=0}^{i-1} \left(1 - \frac{k-1-j}{d}\right)\gamma \\ &= \gamma\left[i\left(1 - \frac{k-1}{d}\right) + \frac{i(i-1)}{2d}\right] \\ &= \gamma i \frac{2d - 2k + i + 1}{2d}, \\ &= \gamma g(i), \end{aligned}$$

and

$$\begin{aligned}
&\sum_{j=0}^{i} b_j + (k-i-1)b_i \\
=&\gamma(i+1)\frac{2d-2k+i+2}{2d} + (k-i-1)\gamma\left(1-\frac{k-1-i}{d}\right) \\
=&\gamma\frac{2ik-i^2-i+2k+2kd-2k^2}{2d}, \\
=&\gamma\frac{B}{f(i)},
\end{aligned}$$

where $f(i)$ and $g(i)$ are defined in (3.2)(3.3). Hence we have:

$$\alpha* = \frac{B-g(i)}{k-i}, \quad \text{for } B \in \left(\frac{\gamma B}{f(i-1)}, \frac{\gamma B}{f(i)}\right].$$

The expression of $\alpha^*(d,\gamma)$ then follows. ■

Chapter 4

Geographic Gossip

4.1 Introduction

In this chapter we go beyond the distributed storage problems and into addressing issues of distributed processing. Motivated by sensor network applications, we consider a network of n nodes, in which each node collects a measurement in some modality of interest (e.g., temperature, light, humidity). In such a setting, it is frequently of interest to solve the *distributed averaging problem*: namely, to develop a distributed algorithm by which all nodes can compute the average of the n sensor measurements. This problem and its connection to Markov chain mixing rates has been studied for over thirty years [deGroot, 1974; Tsitsiklis, 1984]. It has been the focus of renewed interest over the past several years, motivated by various applications in sensor networks and distributed control systems. Early work [deGroot, 1974] studied deterministic protocols, known as consensus algorithms, in which each node communicates with each of its neighbors in every round. More recent work (e.g. [Kempe *et al.*, 2003; Boyd *et al.*, 2004]) has focused on so-called gossip algorithms, a class of randomized algorithms that solve the averaging problem by computing a sequence of pairwise averages. In each round, one node is chosen randomly, and it chooses one of its neighbors randomly. Both nodes compute the average of their values and replace their own value with this average. By iterating this pairwise averaging process, the estimates of all nodes converge to the global average under suitable conditions on the graph topology.

The averaging problem is an archetypal instance of distributed signal processing, in which the goal is to achieve a global objective (e.g., computing the global average of all observations) based on purely local computations (in this case, message-passing between pairs of adjacent nodes). Although distributed averaging itself is a very specialized problem, effective averaging problems provide a useful building block for solving more complex problems in distributed signal processing. Indeed, any averaging

algorithm can be easily converted into a general algorithm that computes any linear projection of the sensor measurements, assuming that each sensor knows the corresponding coefficient of the projection vector. Recently, such algorithms have been proposed for various problems of distributed computation in sensor networks, including distributed filtering, detection, optimization, and compression [Spanos *et al.*, 2005; Xiao *et al.*, 2005; Saligrama *et al.*, 2006; Rabbat *et al.*, 2006a].

A fundamental issue—and the primary focus of this chapter—is how many iterations it takes for any gossip algorithm to converge to a sufficiently accurate estimate. These convergence rates have received significant attention in recent work [Karp *et al.*, 2000; Kempe *et al.*, 2003; Boyd *et al.*, 2004; Boyd *et al.*, 2005; Chen and G. Pandurangan, 2005; Moallemi and van Roy, 2006; Mosk-Aoyama and Shah, 2005; Alanyali *et al.*, 2006]. The convergence speed of a nearest-neighbor gossip algorithm, known as the *averaging time*, turns out to be closely linked to the *spectral gap* (and hence the mixing time) of a Markov matrix defined by a weighted random walk on the graph. Boyd et al. [Boyd *et al.*, 2005] showed how to optimize the neighbor selection probabilities for each node so as to find the fastest-mixing Markov chain on the graph. For certain types of graphs, including complete graphs, expander graphs and peer-to-peer networks, such Markov chains are rapidly mixing, so that gossip algorithms converge very quickly.

Unfortunately, for the graphs corresponding to typical wireless sensor networks, even an optimized gossip algorithm can result in very high energy consumption. For example, a common model for a wireless sensor network is a random geometric graph [Penrose, 2003], in which all nodes are placed uniformly at random in an area and can communicate with neighbors within some fixed radius $r > 0$. With the transmission radius scaling in the standard way [Penrose, 2003] as $r(n) = \Theta(\sqrt{\frac{\log n}{n}})$, even an optimized gossip algorithm requires $\Theta(n^2)$ transmissions (see Section 4.2.4), which is of the same order as the energy required for every node to flood its value to all other nodes. This problem is noted by Boyd et al. [Boyd *et al.*, 2005]: "In a wireless sensor network, Theorem 6 suggests that for a small radius of transmission, even the fastest averaging algorithm converges slowly", and this limitation is intrinsic to standard gossip algorithms applied to such graphs. Intuitively, the nodes in a standard gossip protocol are essentially "blind," and they repeatedly compute pairwise averages with their one-hop neighbors. Information diffuses slowly throughout the network—roughly moving distance $\sqrt{k}$ in k iterations—as in a random walk.

Accordingly, our goal is to develop and analyze alternative—and ultimately more efficient—methods for solving distributed averaging problems in wireless networks. We leverage the fact that sensor nodes typically know their locations, and can exploit this knowledge to perform geographic routing. Localization is itself a well-studied problem (e.g., [Langendoen and Reijers, 2003; He *et al.*, 2003]), since geographic

knowledge is required in numerous applications. With this perspective in mind, we propose an algorithm that, like a standard gossiping protocol, is randomized and distributed, but requires substantially less communication by exploiting geographic information. The idea is that instead of exchanging information with one-hop neighbors, geographic routing can be used to gossip with random nodes who are far away in the network. The bulk of our technical analysis is devoted to showing that the resulting rapid diffusion of information more than compensates for the extra cost of this multi-hop routing procedure.

In effect, routing to far away neighbors creates an overlay communication network that is the complete graph, where an edge is assigned a cost equal to the number of hops on the route between the two nodes. For graphs with regular topology, it is relatively straightforward to see how this additional cost is offset by the benefit of faster convergence time. Indeed, two such examples, the cycle and the grid, are analyzed in Section 5.3, where we show gains of the order n and $\sqrt{n}$ respectively. The more surprising result in chapter is that, by using a simple resampling technique, this type of benefit extends to random geometric graphs—a class of networks with irregular topology that are commonly used as a model of sensor networks formed by random deployments.

This chapter is organized as follows. In Section 5.3, we provide a precise statement of the distributed averaging problem, describe our algorithm, state our main results on its performance, and compare them to previous results in the literature. In Section 4.3, we analyze the performance of our algorithms on two simple regular network topologies, the cycle and the grid. Section 4.4 provides the proofs of our result for the random geometric graph model. In Section 4.5, we provide a number of experimental results that illustrate and complement our theoretical analysis.

4.2 Problem formulation and main results

In this section, we first formulate the distributed averaging problem in sensor networks and then describe our algorithm and main analytical results. We conclude with an overview and comparison to related work.

4.2.1 Problem statement

We begin by formulating the problem of distributed averaging and specifying the technical details of our time and communication models.

4.2.1.1 Distributed averaging

Consider a graph G with vertex set $V = \{1, \ldots, n\}$ and edge set $E \subset V \times V$. Suppose that at time $k = 0$, each node $s \in V$ is given a real-valued number $x_s(0) \in \mathbb{R}$, representing an observation of some type. The goal of distributed averaging is to compute the average $\bar{x}_{\text{ave}} := \frac{1}{n}\sum_{s=1}^{n} x_s(0)$ at *all nodes* of the graph. Consensus and gossip algorithms achieve this goal as follows: at each time slot $k = 0, 1, 2 \ldots$, each node $s = 1, \ldots, n$ maintains an estimate $x_s(k)$ of the global average. We use $x(k)$ to denote the n-vector of these estimates; note that that the estimate at different nodes need not agree (i.e., $x_s(k)$ is in general different from $x_t(k)$ for $s \neq t$). The ultimate goal is to drive the estimate $x(k)$ to the vector of averages $\bar{x}_{\text{ave}}\vec{1}$, where $\vec{1}$ is an n-vector of ones.

For the algorithms of interest to us, the quantity $x(k)$ for $k > 0$ is a random vector, since the algorithms are randomized in their behavior. Accordingly, we measure the convergence of $x(k)$ to $x(0)$ in the following sense [Kempe *et al.*, 2003; Boyd *et al.*, 2005]:

Definition 1. *Given $\epsilon > 0$, the ϵ-averaging time is the earliest time at which the vector $x(k)$ is ϵ close to the normalized true average with probability greater than $1 - \epsilon$:*

$$T_{\text{ave}}(n, \epsilon) = \sup_{x(0)} \inf_{k=0,1,2\ldots} \left\{ \mathbb{P}\left(\frac{\|x(k) - x_{ave}\vec{1}\|}{\|x(0)\|} \geq \epsilon \right) \leq \epsilon \right\}, \tag{4.1}$$

where $\|\cdot\|$ denotes the ℓ_2 norm. Note that this is essentially measuring a rate of convergence in probability.

We can also generalize the pairwise averaging schemes for arbitrary subsets of communicating nodes. At each time-slot k, a random set $S(k)$ of nodes communicate with each other and update their estimates to the average of the estimates of $S(k)$: for all $j \in S(k)$, $x_j(k+1) = \sum_{i \in S(k)} x_i(t)/|S(k)|$. In standard gossip (nearest neighbor) and in geographic gossip, only random pairs of nodes average their estimates, hence $S(k)$ always contains exactly two nodes. On the other hand, in path averaging, $S(k)$ is the set of nodes in the random route generated at each time-slot k. Therefore in this case, $S(k)$ contains a random number of nodes.

4.2.1.2 Asynchronous time model

We use the asynchronous time model [Bertsekas and Tsitsiklis, 1997; Boyd *et al.*, 2005], which is well-matched to the distributed nature of sensor networks. In particular, we assume that each sensor has an independent clock whose "ticks" are distributed

as a rate λ Poisson process. The inter-tick times are exponentially distributed, independent across nodes, and independent across time. We note that this model can be equivalently formulated in terms of a single global clock ticking according to a rate $n\lambda$ Poisson process. By letting Z_k denote the arrival times for this global clock, then the individual clocks can be generated from the global clock by randomly assigning each Z_k to the sensors according to a uniform distribution. On average, there are approximately n global clock ticks per unit of absolute time (an exact analysis can be found in [Boyd *et al.*, 2005]). However, our analysis is based on measuring time in terms of the number of ticks of this (virtual) global clock. Time is discretized, and the interval $[Z_k, Z_{k+1})$ corresponds to the kth timeslot.

Note that throughout this thesis, we are interested in minimizing the number of messages without worrying about delay. We can therefore adjust the length of the timeslots relative to the communication time so that only one packet exists in the network at each timeslot with high probability. Note that this assumption is made only for analytical convenience; in a practical implementation, several packets might co-exist in the network, but the associated congestion control issues are beyond the scope of this work.

4.2.1.3 Communication cost

We compare algorithms in terms of the amount of communication required. We will assume a fixed communication radius and hence the number of one-hop radio transmissions is proportional to the total energy spent for communication. More specifically, let $R(k)$ represent the number of one-hop radio transmissions required for a given node to communicate with some other node in the interval $[Z_k, Z_{k+1})$. In a standard gossip protocol, the quantity $R(k) \equiv R$ is simply a constant, whereas for our protocol, $R(k)$ will be a random variable (with identical distribution for each time slot). The total communication cost, measured in one-hop transmissions, is given by the random variable

$$\mathcal{C}(n, \epsilon) = \sum_{k=1}^{T_{\text{ave}}(n,\epsilon)} R(k) \ . \tag{4.2}$$

We analyze mainly the expected communication cost, denoted by $\mathcal{E}(n, \epsilon)$, which is given by

$$\mathcal{E}(n, \epsilon) = E[R(k)]T_{\text{ave}}(n, \epsilon) \ . \tag{4.3}$$

Our analysis also yields probabilistic upper bounds on the communication cost $\mathcal{C}(n,\epsilon)$ of the form

$$\mathbb{P}\Big\{\mathcal{C}(n,\epsilon) \geq f(n,\epsilon)\Big\} \leq \frac{\epsilon}{2}\ . \tag{4.4}$$

A related convergence metric is consensus time [Denantes *et al.*, 2008] the time T_c required for the error to be divided by an e factor in the long run.

The estimate vector $x(k)$ and the error vector $\varepsilon(k) = x(k) - \bar{x}_{\text{ave}}\vec{1}$ for $k > 0$ are or course random. However, in the long run, the error decays exponentially with a *deterministic* rate $1/T_c$, where T_c, called consensus time, Apart from giving an almost sure criterion for convergence time, consensus time T_c also lightens the formalism by removing the ϵ's (see also [Fagnani and Zampieri, 2008] for a related analysis). Consensus time is formally defined as follows [Denantes *et al.*, 2008]:

Theorem 1. Consensus time T_c. *If $\{S(k)\}_{t\geqslant 0}$ is an independently and identically distributed (i.i.d.) process, then the limit*

$$-\frac{1}{T_c} = \lim_{t\to\infty} \frac{1}{t} \log \|\varepsilon(k)\|, \tag{4.5}$$

where $\|\cdot\|$ denotes the ℓ_2 norm, exists and is a constant with probability 1.

In other words, after a transient regime, the number of iterations needed to reduce the error $\|\varepsilon\|$ by a factor e is almost surely equal to T_c, which therefore characterizes the speed of convergence of the algorithm. T_c is easy to measure in experiments, and can be theoretically upper bounded. However lower bounding this quantity remains an open problem.

Although $T_{ave}(\epsilon)$ is hard to measure in practice, it is easily upper and lower bounded theoretically in terms of the spectral gap (see Section 5.4). Indeed $T_{ave}(\epsilon)$ contains a probability tolerance ϵ in its definition, which simplifies the analysis. An important issue is the behavior of T_c and T_{ave} as the number n of nodes in the network grows. It can be shown that $T_c(n) = O(T_{ave}(n,\epsilon))$ for any fixed ϵ, but whether the two quantities are equivalent and under which conditions is still an open problem. Using the asymptotic consensus time concept, an asymptotic consensus cost $\mathcal{C}_c$ can be defined as follows [Denantes *et al.*, 2008]:

Theorem 2. Consensus cost $\mathcal{C}_c$. *If the averaged sets $\{S(k)\}_{t\geqslant 0}$ are selected in an independent and identically distributed manner, then the following limit exists and is*

a constant with probability 1*:*

$$\begin{aligned} -\frac{1}{\mathcal{C}_c} &= \lim_{t\to\infty} \frac{1}{C(t)} \log \|\varepsilon(t)\| \\ &= \lim_{t\to\infty} \frac{t}{C(t)} \lim_{t\to\infty} \frac{\log \|\varepsilon(t)\|}{t}. \end{aligned}$$

Thus, $\mathcal{C}_c = \mathsf{E}[R(k)]T_c$ *is the number of one-hop transmissions needed in the long run to reduce the error by a factor* e *with probability* 1 *and can be directly bounded by the expected communication cost*

$$\mathcal{C}_c(n) = O(\mathcal{E}(\epsilon, n)). \tag{4.6}$$

for any fixed ϵ*.*

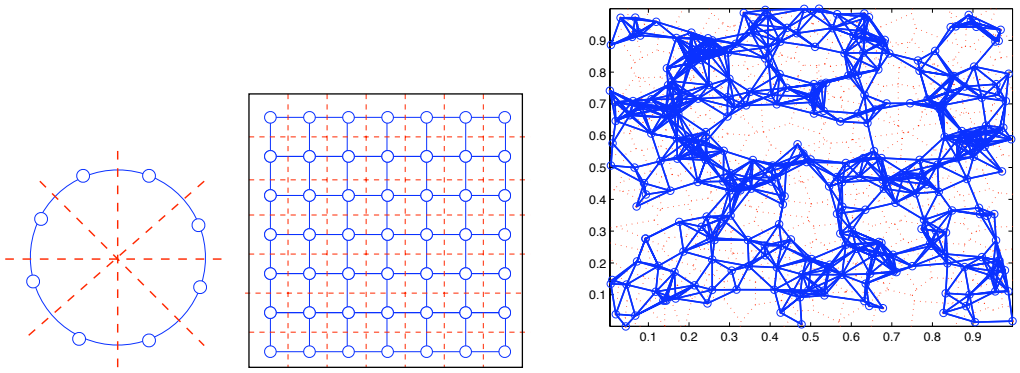
4.2.1.4 Graph topologies

This chapter treats both standard graphs with regular topology, including the single cycle graph and regular grid as illustrated in panels (a) and (b) respectively of Figure 4.1, and an important subclass of random graphs with irregular topologies, namely those formed by random geometric graphs [Penrose, 2003]. The random graph model has been used in previous work on wireless sensor networks [Gupta and Kumar, 2000; Boyd *et al.*, 2005]. More precisely, the random geometric graph $G(n, r)$ is formed by choosing n sensor locations uniformly and independently in the unit square, with any pair of nodes s and t is connected if and only if their Euclidean distance is smaller than some transmission radius r. A sample from this random graph model is illustrated in Figure 4.1(c). It is well known [Penrose, 2003; Gupta and Kumar, 2000; Gamal *et al.*, 2004] that in order to maintain connectivity and minimize interference, the transmission radius $r(n)$ should scale like $\Theta(\sqrt{\frac{\log n}{n}})$. For the purposes of analysis, we assume that communication within this transmission radius always succeeds.[1] Note that we assume that the messages involve real numbers; the effects of message quantization in gossip and consensus algorithms, is an active area of research (see for example [Nedic *et al.*, 2007; Aysal *et al.*, 2008]).

4.2.2 Proposed Algorithm

The proposed algorithm combines gossip with geographic routing. The key assumption is that each node s knows its own geographic location within some compact

[1] However, we note that our proposed algorithm remains robust to communication and node failures.



(a) Cycle (b) Grid (c) Random Geometric Graph

Figure 4.1: Illustration of a various graphs (nodes as circles and edges as solid lines) and the associated Voronoi regions associated with each node (dotted lines). (a) Cycle graph. (b) Regular grid. (c) Random geometric graph.

subset $C \subset \mathbb{R}^2$, specified as a Euclidean pair $(x_s, y_s) \in C$. For the regular grid and random geometric graphs, we take C to be the unit square $[0,1] \times [0,1]$, whereas for the single cycle graph we take C to be the unit circle S^1. In addition, each node can learn the geographic locations of its one-hop neighbors (i.e., vertices $t \in V$ such that $(s,t) \in E$) using a single transmission per node.

Geographic Gossip Algorithm: Suppose the j-th clock tick Z_j is assigned to node s at location $\ell(s)$. The following events then happen:

(1) Node s activates and chooses a point $y = (y_1, y_2)$ uniformly in the region C, referred to as the target location. Node s forms the tuple $m_s = (x_s(j), \ell(s), y)$.

(2) Node s sends m_s to its one-hop neighbor $t \in N(s)$ closest to location y. This operation continues in a recursive manner: when a successive node r receives a packet m_s, it relays the packet m_s to its one-hop neighbor closest to location y. Greedy geographic routing terminates when a node receives the packet and has no one-hop neighbors with distance smaller to the random target that its own. Let v be the node closest to location y.

(3) Node v makes an independent randomized decision to accept m_s. If the packet is accepted, v computes its new value $x_v(j+1) = \frac{1}{2}(x_v(j) + x_s(j))$ and generates a message $m_v = (x_v(j), \ell(v), \ell(s))$, which is sent back to s via greedy geographic routing. Node s can then compute its new value $x_s(j+1) = (x_v(j) + x_s(j))/2$,

and the round ends. If the packet is rejected, then v sends a rejection message to s.

(4) If v rejects the packet from s, then v chooses a new point y' uniformly in the plane and repeats steps (4.2.2)–(4.2.2) with message $m'_s = (x_s(j), \ell(s), y')$.

At a high level, the motivation of the geographic gossip algorithm is to exploit geographic information (via the greedy routing protocol described in step (2)) to create a new communication graph $G' = (V, E')$ as an overlay of the original graph $G = (V, E)$. Note that the new communication graph G' has the same vertex set, but an expanded edge set (i.e., $E' \supset E$). In fact, for all of the versions of geographic gossip analyzed in this chapter, the extended communication graph G' is the complete graph, meaning that $(s, t) \in E'$ for all $s \neq t$. In the standard gossip protocol, each gossip round takes two radio transmissions. In the new communication graph G', certain edges are more costly in terms of one-hop radio transmissions because of the routing required to carry out the communication. On the other hand, the benefit is that the new communication graph G' is dense, so that gossiping converges more quickly. Our main result shows that this tradeoff—between the cost of each gossip round and the total number of rounds—can lead to favorable reductions in the total number of one-hop radio transmissions.

4.2.3 Main Results

The geographic gossip algorithm is a randomized procedure that induces a probability distribution over the sensor v chosen at each round. By construction, the probability of choosing sensor v in step (2) of the geographic gossip algorithm is equal to a_v, the area of its associated Voronoi region. For certain types of regular graphs, such as the single cycle and regular grid shown in panels (a) and (b) of Figure 4.1, this distribution over Voronoi regions is uniform. In this particularly favorable setting, the "randomized" decision of node v in step (3) is simple: it accepts the packet m_s with probability one. With this choice, the distribution over chosen nodes v is guaranteed to be uniform for these regular graphs. Consequently, it can be shown using known results for mixing on the complete graph that the averaging time of geographic gossip $T_{\text{ave}}(n, \epsilon)$ is $O(n \log \epsilon^{-1})$. The communication cost given by $\mathcal{E}(n, \epsilon) = \mathbb{E}[R(k)]T_{\text{ave}}(n, \epsilon)$, where $R(k) \equiv R$ is the number of single-hop communications required in round k of the protocol. By computing the expected value $\mathbb{E}[R]$, it can be shown that the overall communication costs for these regular topologies scale as $\mathcal{E}(n, \epsilon) = \Theta(n^2 \log \epsilon^{-1})$ for the single cycle, and $\mathcal{E}(n, \epsilon) = \Theta(n^{1.5} \log \epsilon^{-1})$ for the regular grid. Thus, as derived in Section 4.3, geographic gossip yields improvements by factors of n and $\sqrt{n}$ over standard gossip for these regular graphs.

For random geographic graphs, in contrast, the distribution of Voronoi regions is quite non-uniform. Consequently, in order to bound the averaging time $T_{\text{ave}}(n, \epsilon)$, we use in step (3) a rejection sampling scheme previously proposed by Bash et al. [Bash *et al.*, 2004] in order to "temper" the distribution. Given the n-vector $\vec{a}$ of areas of the sensors' Voronoi regions, we set a threshold τ. Sensors with cell area smaller than τ always accept a query, and sensors with cell areas larger than τ may reject the query with a certain probability. The rejection sampling method simultaneously protects against oversampling and limits the number of undersampled sensors, which allows us to prove that $T_{\text{ave}}(n, \epsilon) = O(n \log \epsilon^{-1})$ even for this perturbed distribution.

Of course, nothing comes for free: the rejection sampling scheme requires a random number Q of queries before a sensor accepts. Since the queries are independent, Q is a geometric random variable with parameter equal to the probability of a query being accepted. In terms of the number of queries, the total number of radio transmissions for the kth gossip round is $R(k) = O(Q \cdot G)$. Therefore if T_{ave} gossip rounds take place overall, the expected of radio transmissions will be $\mathcal{E}(n, \epsilon) = \mathbb{E}[Q \cdot G \cdot T_{\text{ave}}(n, \epsilon)]$. Accordingly, a third key component of our analysis in Section 4.4 is to show that the probability of acceptance remains *larger than a constant*, which allows us to upper bound the expectation of the geometric random variable Q by a constant. We also establish an upper bound on the maximum value of Q over T_{ave} rounds that holds with probability greater than $1 - \epsilon/2$.

Putting together the pieces yields our main result for random geometric graphs: the expected cost for computing the average with the proposed geographic gossip algorithm is

$$\mathcal{E}(n, \epsilon) = O\left(\frac{n^{3/2}}{\sqrt{\log n}} \log \epsilon^{-1}\right). \tag{4.7}$$

In comparison to previous results on standard gossip for random graphs [Boyd *et al.*, 2005], geographic gossip yields a reduction by a factor of $\sqrt{\frac{n}{\log n}}$ in the number of one-hop communication rounds.

We note for some classes of graphs, the rejection sampling may not be necessary, even when the induced distribution is not uniform, as long as it is reasonably close to uniform. In particular, if we have a $\Omega(n^{-1})$ lower bound on the area of a Voronoi cell for all sensors, then sampling by area is approximately uniform. If we can obtain a slightly looser bound on the deviations of the Voronoi areas, alternative techniques may be able to show that our algorithm will not suffer a performance loss without rejection sampling. However, for geometric random graphs, it is difficult to obtain a good lower bound on the Voronoi cell size, which is our motivation for applying and analyzing the rejection sampling scheme.

4.2.4 Related work and comparisons

Boyd et al. [Boyd *et al.*, 2005; Boyd *et al.*, 2004] have analyzed the performance of standard gossip algorithms. Their fastest standard gossip algorithm for the ensemble of random geometric graphs $G(n, r)$ has a ϵ-averaging time [Boyd *et al.*, 2005] $T_{\text{ave}}(n, \epsilon) = \Theta(n\frac{\log \epsilon^{-1}}{r(n)^2})$. (This quantity is computed in section IV.A of Boyd et al. [Boyd *et al.*, 2005] but the result is expressed in terms of absolute time units which needs to be multiplied by n to become clock ticks). Consequently, for the standard choice of radius $r(n) = \Theta(\sqrt{\frac{\log n}{n}})$ ensuring network connectivity, this averaging time scales as $\Theta(\frac{n^2}{\log n} \log \epsilon^{-1})$. In standard gossip, each gossip round corresponds to communication with only one-hop neighbor and hence costs only one radio transmission which means that the fastest standard gossip algorithm will have a total cost $\mathcal{E}(n) = \Theta(\frac{n^2}{\log n} \log 1/\epsilon)$ radio transmissions. Therefore, our proposed algorithm saves a factor of $\sqrt{\frac{n}{\log n}}$ in communication energy by exploiting geographic information.

A number of recent papers [Moallemi and van Roy, 2006; Mosk-Aoyama and Shah, 2005; Alanyali *et al.*, 2006] have also considered the problem of computing averages in networks. The consensus propagation algorithm of Moallemi and van Roy [Moallemi and van Roy, 2006] is a modified form of belief propagation that attempts to mitigate the inefficiencies introduced by the "random walk" in gossip algorithms. For the single cycle graph, they show improvement by a factor of $\Theta(\frac{n}{\log n})$ over standard gossip. Our results for geographic gossip on the single cycle (see Section 4.3) show improvement by a factor of $\Theta(n)$ over standard gossip, and hence a factor $\Theta(\log n)$ over consensus propagation. Mosk-Aoyama and Shah [Mosk-Aoyama and Shah, 2005] use an algorithm based on Flajolet and Martin [Flajolet and Martin, 1985] to compute averages, and bound the averaging time in terms of a "spreading time" associated with the communication graph. However, they only show the optimality of their algorithm for a graph consisting of a single cycle, so it is currently difficult to speculate how it would perform on other regular graphs or geometric random graphs. Alanyali et al. [Alanyali *et al.*, 2006] consider the related problem of computing the average of a network at a *single* node (in contrast to computing the average in parallel at every node). They propose a distributed algorithm to solve this problem and show how it can be related to cover times of random walks on graphs.

Li and Dai [Li and Dai, 2008] recently proposed Location-Aided Distributed Averaging (LADA), a scheme that uses partial locations and markov chain lifting to create fast gossiping algorithms. The cluster-based LADA algorithm performs slightly better than geographic gossip, requiring $\Theta(n^{1.5} \log \epsilon^{-1}/(\log n)^{1.5})$ messages for random geometric graphs. While the theoretical machinery is different, LADA algorithms also use directionality to accelerate gossip, but can operate even with partial location

information and have smaller total delay compared to geographic gossip, at the cost of a somewhat more complicated algorithm.

4.3 Analysis for Regular Networks

In this section, we illustrate the benefits of our geographic gossip algorithm for two simple networks, the ring and the grid, both of which are regular graphs. Due to this regularity, the implementation and analysis of geographic gossip turns out to be especially simple. More specifically, when these graphs are viewed as contained with the unit disk (ring graph) or the unit square (grid graph), then the Voronoi region of each node is equal in area (see Figure 4.1). Consequently, sampling a *location* uniformly in the space is equivalent to sampling a *sensor* uniformly, and thus the overlay graph created by geographic routing (step (2) of the geographic gossip algorithm) is a complete graph with uniform edge weights. In this case, the randomized decision rule in step (4.2.2) is not needed — the target v always accepts the message. For the ring, we show that standard gossip has a communication cost $\mathcal{E}(n, \epsilon)$ for ϵ-accuracy that scales as $\Theta(n^3 \log \epsilon^{-1})$, and that geographic gossip can improve this to $O(n^2 \log \epsilon^{-1})$. For the grid, we show that standard gossip has communication cost $\Theta(n^2 \log \epsilon^{-1})$, and geographic gossip can improve this to $O(n^{3/2} \log \epsilon^{-1})$.

4.3.1 Analysis of single cycle graph

The ring network consists of a single cycle of n nodes equispaced on the unit circle (see Figure 4.1(a)). For this simple network, we have the following result characterizing the improvement of geographic gossip over standard gossip:

Proposition 1. *In terms of the communication cost $\mathcal{E}(n, \epsilon)$ for ϵ-accuracy, geographic gossip yields a $\Omega(n)$ improvement over standard gossip on the single cycle graph.*

Proof. We first compute the communication cost $\mathcal{E}(n, \epsilon)$ for standard gossip. In standard nearest-neighbor gossip, the probability p_{ij} that nodes i chooses to average with node j is 0 unless $|i - j| = 1$, otherwise it is $1/2$. Therefore the matrix $P = (p_{ij})$ is a symmetric circulant matrix, generated by the n-vector $(0, 1/2, 0, 0, \ldots, 1/2)$. Using previous results on standard gossip [Boyd *et al.*, 2005], in order to evaluate the performance of standard gossip, we must find the second eigenvalue λ_2 of the matrix W

defined by

$$D = \text{diag}\left(\left\{\sum_{j=1}^{n}(P_{ij}+P_{ji}) : i = 1,2,\ldots,n\right\}\right) \;=\; 2I$$
$$W = I + \frac{1}{2n}D + \frac{1}{2n}(P+P^T) \;=\; \left(1-\frac{1}{n}\right)I + \frac{1}{n}P\ .$$

Note that W is also a circulant matrix, generated by the n-vector $(1-n^{-1}, (2n)^{-1}, 0, 0, \ldots, (2n)^{-1})$. Circulant matrices are diagonalized by the discrete Fourier Transform (DFT) matrix, so that the eigenvalues can be computed explicitly as

$$\left(1-\frac{1}{n}\right) + \frac{1}{n}\cos\frac{k2\pi}{n} \quad k = 0,2,\ldots,n-1\ .$$

Consequently, the second largest eigenvalue is given by

$$\lambda_2(W) = 1 + \frac{1}{n}\sum_{j=1}^{\infty}(-1)^{2j}\frac{1}{(2j)!}\left(\frac{2\pi}{n}\right)^{(2j)} \;=\; 1+\Theta(n^{-3})\ .$$

Therefore, by a Taylor series expansion, we have $\log\lambda_2(W) = \Theta(n^{-3})$. Applying previous results [Boyd *et al.*, 2005] on standard gossip, we conclude that the ϵ-averaging time of standard gossip is:

$$T_{\text{ave}}(n,\epsilon) = \Theta\left(\frac{\log\epsilon^{-1}}{\log\lambda_2(W)^{-1}}\right) \;=\; \Theta\left(n^3\log\epsilon^{-1}\right)$$

Since each gossip communication costs us one hop, the average number of one-hop transmissions for standard gossip on the ring is

$$\mathcal{E}(n,\epsilon) = \Theta\left(n^3\log\epsilon^{-1}\right). \tag{4.8}$$

We now show how geographic gossip reduces the number of one-hop transmissions. In geographic gossip for the ring network, a source node chooses a random location within the unit circle uniformly at random, which induces a uniform distribution over the nodes in the network (see Figure 4.1(a)). It then sends a packet to its target around the ring and they exchange values. We think of geographic gossip as running a gossip algorithm on the complete graph with $p_{ij} = n^{-1}$ for all i and j. For this

graph, we have

$$W = \left(1 - \frac{1}{n}\right) I + \frac{1}{n^2} \vec{1}\,\vec{1}^T.$$

Calculating the second largest eigenvalue yields $\lambda_2(W) = 1 - \frac{1}{n} + \frac{1}{n^2} = 1 - \Theta(n^{-1})$, so $\log \lambda_2(W) = \Theta(n^{-1})$, and hence $T_{\text{ave}}(n, \epsilon) = \Theta\left(n \log \epsilon^{-1}\right)$. By summing over the pairwise distances in the graph, we see that the expected number of one-hop transmissions at any round is bounded by

$$\mathbb{E}[R] \;=\; \mathbb{E}[R(k)] \leq \frac{1}{n} \sum_{i=1}^{\lceil \frac{n}{2} \rceil} (2) = O(n)\ .$$

Thus, the expected number of transmissions for geographic gossip is given by

$$\mathcal{E}(n, \epsilon) = T_{\text{ave}}(n, \epsilon)\, \mathbb{E}[R] \;=\; O\left(n^2 \log \epsilon^{-1}\right)\ . \tag{4.9}$$

Comparing equations (4.8) and (4.9) yields the claim. ■

As demonstrated by this result, for the ring network, using geographic knowledge and routing improves the energy consumption as measure in hops by a factor of n. In standard gossip, information from one node diffuses slowly in a ring, taking almost n^2 steps to become uniformly distributed. Geographic gossip allows the information from one node in the network to travel larger distances at the expense of the routing cost.

4.3.2 Analysis of regular grid

We now turn to geographic gossip on the two dimensional grid defined by a collection of n vertices s_{ij} located at positions $(i/\sqrt{n}, j/\sqrt{n})$ within the unit square $[0, 1] \times [0, 1]$, as illustrated in Figure 4.1(c).

Proposition 2. *In terms of the communication cost $\mathcal{E}(n, \epsilon)$ required to achieve ϵ-accuracy, geographic gossip yields a $\Omega(\sqrt{n})$ improvement over standard gossip on the regular 2-D grid.*

Proof. The performance of standard gossip on the grid can be calculated using Corollary 1 from Boyd et al. [Boyd *et al.*, 2006], which says that the averaging time is given by $T_{\text{ave}}(n, \epsilon) = \Theta\left(\frac{n \log \epsilon^{-1}}{1 - \lambda_2(P)}\right)$. For standard gossip on the grid, the matrix P is simply the transition matrix of a random walk on the two-dimensional grid, for which it is

known [Aldous and Fill, 2007] that $(1 - \lambda_2(P))^{-1} = \Theta(n)$. Consequently, we have $T_{\text{ave}}(n, \epsilon) = \Theta\left(n^2 \log \epsilon^{-1}\right)$, so that the average number of one-hop transmissions is

$$\mathcal{E}(n, \epsilon) = \Theta\left(n^2 \log \epsilon^{-1}\right) . \tag{4.10}$$

Now let us turn to geographic gossip. For a regular topology like the grid, the Voronoi cells are all of equal area, so in step (4.2.2) of the geographic gossip algorithm, the chosen target v simply accepts with probability one. Consequently, the number of one-hop communications per round is simply the route length. For a regular 2-dimensional grid, routing the message at round k costs $\mathbb{E}[R(k)] = O(\sqrt{n})$ one-hop transmissions. As we derived for the ring network, the geographic gossip algorithm is communicating on an overlay network that is fully connected, so that the number of rounds required scales as $T_{\text{ave}}(n, \epsilon) = O\left(n \log \epsilon^{-1}\right)$. Putting the pieces together, we conclude that the total communication cost for ϵ-accuracy using geographic gossip scales as

$$\mathcal{E}(n, \epsilon) = O\left(n^{3/2} \log \epsilon^{-1}\right) . \tag{4.11}$$

Comparing equations (4.10) and (4.11) yields the claim. ■

Thus, for the regular grid in 2-dimensions, geographic gossip yields a factor of $\sqrt{n}$ savings in the convergence time. The ease of our analysis in both of the preceding examples—ring and grid networks—arises from the regularity of the topology, which allowed us to either write the transition matrix explicitly or use standard results. The following section is devoted to analysis of geographic gossip for random geometric graphs, where we will derive a similar performance improvement. For random geometric graphs, in contrast to the regular topologies considered thus far, we will use a non-trivial randomized decision rule in step (4.2.2) of the gossip algorithm in order to compensate for irregularities of the graph topology and areas of Voronoi regions.

4.4 Analysis for Random Geometric Graphs

We now turn to an analysis of the number of one-hop communications needed for our algorithm in the case of the random geometric graph model. At a high level, our analysis consists of three main steps:

1. First, we address the number of one-hop transmissions G required to route a packet from node s to the randomly chosen target v (see step (2) of the geographic gossip algorithm). We first prove that when the connectivity radius

of the random graphs scales in the standard way as $r(n) = \Theta(\sqrt{\frac{\log n}{n}})$, greedy routing always reaches the closest node v to the random target with

$$G = O\left(\sqrt{\frac{n}{\log n}}\right) \tag{4.12}$$

one-hop radio transmissions. Note that in practice more sophisticated geographic routing algorithms (e.g., [Karp and Kung, 2000]) can be used to ensure that the packet approaches the random target when there are "holes" in the node coverage. However, greedy geographic routing is adequate for the problem considered here.

2. As discussed above, when geographic gossip is applied to a graph with an irregular topology (such as a random geometric graph), it is necessary to compensate for the irregularity with a non-trivial accept/reject protocol in step (3) of the algorithm. Accordingly, our next step is to bound the expected number of rejections experienced by a given sensor s.

3. The final step is to analyze the number of such gossip rounds needed for the average to converge to within the target error.

We take up each of these factors in turn in the subsections to follow.

4.4.1 Routing in $O(1/r(n))$

We first address how to choose the transmission radius of the sensors in order to guarantee the network's connectivity and the success of greedy geographic routing.

Lemma 1 (Network connectivity)**.** *Let a graph be drawn randomly from the geometric ensemble $G(n, r)$ defined in Section 4.2.1, and a partition be made of the unit area into squares of side length $\alpha(n) = \sqrt{2\frac{\log n}{n}}$. Then the following statements all hold with high probability:*

(a) Each square contains at least one node.

(b) If $r(n) = \sqrt{10\frac{\log n}{n}}$, then each node can communicate to a node in the four adjacent squares.

(c) All the nodes in each square are connected with each other.

Proof. The total number of squares of side length $\alpha(n)$ is $M = \frac{n}{2\log n}$. We view these as "bins" into which the n sensors are assigned uniformly. Standard results on this

random process [Motwani and Raghavan, 1995b; Gamal *et al.*, 2004] show that with high probability $\Theta(M \log M)$ sensors are sufficient to cover all of the bins, concluding the proof.

■

Lemma 2 (Greedy geographic routing)**.** *Suppose that a node target location is chosen in the unit square. Then greedy geographic routing routes to the node closest to the target in* $O(1/r(n)) = O(\sqrt{\frac{n}{\log n}})$ *steps.*

Proof. By Lemma 1(a), every square of side length $\alpha(n) = \sqrt{2\frac{\log n}{n}}$ is occupied by at least a node. Therefore, we can perform greedy geographic routing by first matching the row and then the column of the square which contains the target, which requires at most $\frac{2}{r(n)} = O(\sqrt{\frac{n}{\log n}})$ hops. After reaching the square where the target is contained, Lemma 1(c) guarantees that the subgraph contained in the square is completely connected. Therefore, one more hop suffices to reach the node closest to the target.

■

These routing results allow us to bound the cost in hops for an arbitrary pair of nodes in the network to exchange values. In the next section, we analyze a rejection sampling method used to reduce the nonuniformity of the distribution.

4.4.2 Rejection sampling

As mentioned in the previous section, sampling geographic locations uniformly induces a nonuniform sampling distribution on the sensors. Assigning locations to the nearest sensors induces a Voronoi tessellation of the plane, and sensor v is queried with probability proportional to the area a_v of its Voronoi cell. By judiciously rejecting queries, the sensors with larger Voronoi areas can ensure that they are not oversampled. We adopt the rejection sampling scheme proposed by Bash et al. [Bash *et al.*, 2004]: when queried, sensor v *accepts* the request with probability

$$r_v = \min\left(\frac{\tau}{a_v}, 1\right) \ , \tag{4.13}$$

where τ is a predefined threshold. Thus sensors with small Voronoi regions always accept, and sensors with large Voronoi regions sometimes reject.

Given τ, the probability q_v that sensor v is sampled can be written as:

$$\begin{aligned} q_v &= \frac{\min(\tau, a_v)}{\sum_{t=1}^{n} \min(\tau, a_t)} \\ &= \frac{\min(\tau, a_v)}{|\{t : a_t \geq \tau\}| \cdot \tau + \sum_{t:a_t<\tau} a_t} \ . \end{aligned} \tag{4.14}$$

Here the denominator in expression (4.14) is the total chance that a query is accepted:

$$P_a = \sum_{v=1}^{n} a_v \min\left(\frac{\tau}{a_v}, 1\right) = |\{v : a_v \geq \tau\}|\tau + \sum_{v:a_v<\tau} a_v \ . \tag{4.15}$$

Let Q denote the total number of requests made by a sensor before one is accepted.

Rejection sampling "slices" the histogram at τ, and renormalizes the distribution accordingly. The total area that is sliced off is equal to $1 - P_a$, the probability that a query is rejected. Thus, we see that if τ is chosen to be too small, then the probability of rejection becomes very large. Lemma 3 addresses this concern—in particular, by establishing that the choice $\tau = \Theta(n^{-1})$ suffices to keep the rejection probability suitably bounded away from 1, so that the expected number of queries $\mathbb{E}[Q]$ remains finite. More specifically, we choose τ such that

$$\mathbb{P}(a_v \leq \tau) = \min\left(\nu, \frac{\mu}{1+\mu}\right) \ , \tag{4.16}$$

where the constants ν and μ control the undersampling and oversampling respectively. With this choice of τ, the results of Bash et al. [Bash *et al.*, 2004] ensure that no sensor is sampled with probability greater than $(1+\mu)/n$ and no more than νn sensors are sampled with probability less than $1/n$. The following result establishes that the acceptance probability remains sufficiently large:

Lemma 3. *Ler $0 < c < 1/4$. For $\tau = cn^{-1}$, we have $\mathbb{P}(a_v > \tau) \geq 1 - 4c$.*

Proof. We use a simple geometric argument to lower bound $\mathbb{P}(a_v > \tau)$. Consider a node s such that a circle of area τ it lies entirely within its Voronoi region. Clearly, such nodes are a subset of those with area larger than τ. The radius of this circle is $r = \sqrt{\tau/\pi}$. Note that r is no more than half the distance to the nearest node. Thus in order to inscribe a circle of radius τ in the Voronoi region, all other nodes must lie outside a circle of radius $2r$ around the node. This larger circle has area 4τ, so

$$\mathbb{P}(a_v > \tau) \geq (1 - 4\tau)^{n-1} = (1 - 4cn^{-1})^{n-1} \geq 1 - 4c \ . \tag{4.17}$$

Thus, by appropriate choice of c, we can make the acceptance probability arbitrarily

close to 1. ■

Our next step is to bound the distance between the new sampling distribution $\vec{q}$ (i.e., after tempering by the rejection sampling procedure), and the uniform distribution $n^{-1}\vec{1}$ over acceptance regions. These bounds are used in next section to bound an eigenvalue of a matrix associated with the gossip algorithm.

Lemma 4. *For any $\epsilon > 0$, there exists constants $\mu > 0$ and $\nu > 0$ such that rejection sampling with parameters (μ, ν) ensures that*

$$\left\| \vec{q} - \frac{1}{n}\vec{1} \right\|_1 < \epsilon, \qquad \text{and} \tag{4.18a}$$

$$\left\| \vec{q} - \frac{1}{n}\vec{1} \right\|_2 < \frac{1}{\sqrt{n}}\epsilon \ . \tag{4.18b}$$

Proof. Given $\epsilon > 0$, choose ν and μ such that $\nu + \mu < \epsilon$ and $\nu + \mu^2 < \epsilon^2$. We then expand and bound the error function as

$$\sum_{v=1}^{n} \left| q_v - \frac{1}{n} \right| \leq \sum_{v:q_v<1/n} \left| \frac{1}{n} - q_v \right| + \sum_{v:q_v \geq 1/n} \left| q_v - \frac{1}{n} \right| .$$

Now we use the properties of rejection sampling from [Bash *et al.*, 2004]:

$$q_v \leq \frac{1+\mu}{n} \qquad \forall v \tag{4.19}$$

$$\left| \left\{ v : q_v < \frac{1}{n} \right\} \right| \leq \nu n \ . \tag{4.20}$$

On the set $\{v : q_v \geq 1/n\}$ we use the first bound and on the set $\{v : q_v < 1/n\}$ we use the second bound:

$$\begin{aligned} \sum_{v=1}^{n} \left| q_v - \frac{1}{n} \right| &\leq \left(\nu n \frac{1}{n} + n \left(\frac{1+\mu}{n} - \frac{1}{n} \right) \right) \\ &\leq \nu + \mu \ , \end{aligned}$$

which is less than ϵ by our choice of ν and μ.

Turning now to the bound (4.18b), we write

$$\begin{aligned}\left\|\vec{q}-\frac{1}{n}\vec{1}\right\|_2^2 &= \sum_{v:a_v<\tau}\left|q_v-\frac{1}{n}\right|^2+\sum_{v:a_v\geq\tau}\left|q_v-\frac{1}{n}\right|^2\\ &\leq \nu n\frac{1}{n^2}+n\left(\frac{\mu}{n}\right)^2\\ &\leq \frac{1}{n}(\nu+\mu^2)\\ &\leq \frac{1}{n}\epsilon^2\ .\end{aligned}$$

■

Finally, we need to bound the expected number of rejections and the maximum number of rejections in order to bound the expected number of transmissions and total transmission time. Recall that Q is the number of queries that a sensor has to make before one is accepted, and has a geometric distribution:

$$\mathbb{P}(Q=t)=P_a(1-P_a)^t\ . \tag{4.21}$$

Lemma 5. *For a fixed (μ,ν), rejection sampling leads to a constant number of expected rejections.*

Proof. The random variable Q is just a geometric random variable with parameter P_a, so we can write its mean as:

$$\begin{aligned}\mathbb{E}[Q] &= \frac{1}{P_a}\\ &= \frac{1}{|\{v:a_v\geq\tau\}|\tau+\sum_{v:a_v<\tau}a_v}\\ &\leq \frac{1}{(1-\nu)\tau n}\\ &= O(1)\ ,\end{aligned}$$

where the final step follows since $\tau=\Theta(n^{-1})$ by construction. ■

Lemma 6. *Let $\{Q_k : k=1,2,\dots K\}$ be a set of i.i.d. geometric random variables with parameter P_a. For any fixed pair (μ,ν), rejection sampling gives*

$$\max_{1\leq k\leq K}Q_k=O(\log K+\log\epsilon^{-1}) \tag{4.22}$$

with probability greater than $1-\epsilon/2$.

Proof. For any integer $m \geq 2$, a straightforward computation yields that

$$\mathbb{P}(Q \leq m-1) = \sum_{t=0}^{m-1} P_a \left(1-P_a\right)^t = 1 - (1-P_a)^m.$$

By the i.i.d. assumption, we have

$$\begin{aligned}\mathbb{P}(\max_k Q_k \leq m-1) &= \left[1-(1-P_a)^m\right]^K \\ &= \left[1-\exp(m\log(1-P_a))\right]^K \ .\end{aligned}$$

We want to choose $m = m(K,\epsilon)$ such that this probability is greater than or equal to $1-\epsilon/2$. First set $m = -\rho\frac{\log K}{\log(1-P_a)}$, where ρ is to be determined. Then we have

$$\mathbb{P}(\max_k Q_k \leq m-1) = \left[1-1/K^\rho\right]^K \ .$$

We now need to choose $\rho > 1$ such that

$$\left[1-1/K^\rho\right]^K \geq 1-\epsilon/2 \ ,$$

or equivalently, such that

$$1-\left[1-1/K^\rho\right]^K \leq \epsilon/2 \ .$$

Without loss of generality, let K be even. Then by convexity, we have $(1-y)^K \geq 1-Ky$. Applying this with $y = 1/K^\rho$, we obtain

$$1-\left[1-1/K^\rho\right]^K \leq 1/K^{\rho-1}.$$

Hence we need to choose $\rho \geq \log(2/\epsilon)/\log K + 1$ for the bound to hold. Thus, if we set

$$m = -\rho\frac{\log K}{\log(1-P_a)} = O(\log\epsilon^{-1} + \log K) \ ,$$

then with probability greater than $1-\epsilon/2$, all K rounds of the protocol use less than m rounds of rejection. ∎

4.4.3 Averaging with gossip

As with averaging algorithms based on pairwise updates [Boyd *et al.*, 2005], the convergence rate of our method is controlled by the second largest eigenvalue, denoted

$\lambda_2(W)$, of the matrix

$$W := I + \frac{1}{2n}\left[P + P^T - D\right] \ ,$$

where D is diagonal with entries $D_i = (\sum_{j=1}^{n}[P_{ij} + P_{ji}])$. The (i,j)-th entry of the matrix P is the probability that node i exchanges values with node j. Without rejection sampling, $P_{ij} = a_j$, and with rejection sampling, $P_{ij} = q_j$. With this notation, we are now equipped to state and prove the main result of this chapter.

Theorem 3. *The geographic gossip protocol with rejection threshold $\tau = \Theta(n^{-1})$ has an averaging time*

$$T_{\mathrm{ave}}(n,\epsilon) = O\left(n \log \epsilon^{-1}\right) \ . \tag{4.23}$$

Proof. To establish this bound, we exploit Theorem 3 of [Boyd *et al.*, 2005], which states that the ϵ-averaging time is given by

$$T_{\mathrm{ave}}(\epsilon, P) = \Theta\left(\frac{\log \epsilon^{-1}}{\log \lambda_2(W)^{-1}}\right) \ . \tag{4.24}$$

Thus, it suffices to prove that $\log \lambda_2(W) = \Omega(1/n)$ in order to establish the claim.

The probability of any sensor choosing sensor v is just q_v, so that we can write P as the outer product $P = \vec{1}\vec{q}^T$. Note that the diagonal matrix D has entries

$$D_i = \sum_{j=1}^{n}(P_{ij} + P_{ji}) = \sum_{j=1}^{n} q_j + \sum_{j=1}^{n} q_i = 1 + nq_i \ .$$

Overall, we can write W in terms of outer products as:

$$W = \left(I - \mathrm{diag}(\vec{1} + n\vec{q})\right) + \frac{1}{2n}(\vec{1}\,\vec{q}^T + \vec{q}\,\vec{1}^T) \ . \tag{4.25}$$

Note that the matrix W is symmetric and positive semidefinite.

We claim that the second largest eigenvalue $\lambda_2(W) = O(1-c/n)$, for some constant c. By a Taylor series expansion, this implies that $\log \lambda_2(W) = \Theta(n^{-1})$ as desired. To simplify matters, we transform the problem to finding the maximum eigenvalue of an alternative matrix. Since W is doubly stochastic, Perron-Frobenius theory [Horn and Johnson, 1987] guarantees that its largest eigenvalue is one, and has associated eigenvector $v_1 = n^{-1/2}\vec{1}$. Consider the matrix $W' = W - \frac{1}{n^2}\vec{1}\,\vec{1}^T$; using equation (4.25), it can be decomposed as

$$W' = D' + Q',$$

where $D' = (I - (2n)^{-1}\operatorname{diag}(\vec{1} + n\vec{q}))$ is diagonal and

$$Q' = \frac{1}{2n}\left(\vec{1}(\vec{q} - n^{-1}\vec{1})^T + (\vec{q} - n^{-1}\vec{1})\vec{1}^T\right)$$

is symmetric.

Note that by construction, the eigenvalues of W' are simply

$$\lambda(W') = \left\{1 - \frac{1}{n}, \lambda_2(W), \ldots, \lambda_n(W)\right\} .$$

On one hand, suppose that $\lambda_1(W') > \lambda_2(W)$; in this case, then $(1 - \frac{1}{n}) > \lambda_2(W)$ and we are done. Otherwise, we have

$$\lambda_1(W') = \lambda_2(W) .$$

Note that W' is the sum of two Hermitian matrices – a diagonal matrix and a symmetric matrix with small entries. We can therefore apply Weyl's theorem [Horn and Johnson, 1987, p.181], to obtain that

$$\lambda_1(W') \leq \lambda_1(D') + \lambda_1(Q') \leq \left(1 - \frac{1}{2n}\right) + \lambda_1(Q') .$$

It is therefore sufficient to bound $\lambda_1(Q')$. We do so using the Rayleigh-Ritz theorem [Horn and Johnson, 1987, p.176], the Cauchy-Schwartz inequality, and Lemma 4 as follows:

$$\begin{aligned}
\lambda_1(Q') &= \max_{\vec{y}:\|\vec{y}\|_2=1} \vec{y}^T Q' \vec{y} \\
&= \frac{1}{2n} \max_{\vec{y}:\|\vec{y}\|_2=1} \vec{y}^T(\vec{1}(\vec{q} - n^{-1}\vec{1})^T + (\vec{q} - n^{-1}\vec{1})\vec{1}^T\vec{y} \\
&= \frac{1}{n} \max_{\vec{y}:\|\vec{y}\|_2=1} \vec{y}^T\vec{1}(\vec{q} - n^{-1}\vec{1})^T\vec{y} \\
&\leq \frac{1}{n} \max_{\vec{y}:\|\vec{y}\|_2=1} \|\vec{y}\|_2 \cdot \|\vec{1}\|_2 \cdot \|\vec{q} - n^{-1}\vec{1}\|_2 \cdot \|\vec{y}\|_2 \\
&\leq \frac{1}{n}\left(1 \cdot \sqrt{n} \cdot \frac{1}{\sqrt{n}}\epsilon\right) \\
&= \frac{1}{n}\epsilon .
\end{aligned}$$

Overall we have proved the bound

$$\lambda_1(W') \leq \left(1 - \frac{1}{2n}\right) + \frac{1}{n}\epsilon \ . \tag{4.26}$$

We can choose $\epsilon < 1/4$ using Lemma 4 to get the desired bound. ■

The preceding theorem shows that by using rejection sampling we can bound the convergence time of the gossip algorithm. We can therefore bound the number of radio transmissions required to estimate the average.

Corollary 1. *The expected number of radio transmissions required for our gossip protocol on the geometric random graph $G(n, \sqrt{\frac{\log n}{n}})$ is upper bounded by*

$$\mathcal{E}(n, \epsilon) = O\left(\frac{n^{3/2}}{\sqrt{\log n}} \log \epsilon^{-1}\right) \ . \tag{4.27}$$

Moreover, with probability greater than $1 - \epsilon/2$, the maximum number of radio transmissions is upper bounded

$$\mathcal{C}(n, \epsilon) = O\Big(\mathcal{E}(n, \epsilon)\big[\log n + \log \epsilon^{-1}\big]\Big) \ . \tag{4.28}$$

Remark: Note that for $\epsilon = n^{-\alpha}$ for any $\alpha > 0$, our bounds are of the form $\mathcal{E}(n, 1/n^{\alpha}) = O(n^{3/2}\sqrt{\log n})$ and $\mathcal{C}(n, \epsilon) = O(n^{3/2} \log^{3/2} n)$.

Proof. We just have to put the pieces together. If we assume an asynchronous protocol, the cost per transmission pair is given by the product of $O(\sqrt{n/\log n})$ from routing, $\mathbb{E}[Q]$ from rejection sampling, and the averaging time T_{ave}. From Lemma 5, $\mathbb{E}[Q] = O(1)$. Using equation (4.24) and Theorem 3, we can bound $\log \lambda_2(W)^{-1}$ by $(1 - \lambda_2(W)) = O(n^{-1})$. Thus, the expected number of communications is

$$O\left(\sqrt{\frac{n}{\log n}}\mathbb{E}[Q] n \log \epsilon^{-1}\right) = O\left(\frac{n^{3/2}}{\sqrt{\log n}} \log \epsilon^{-1}\right) \ . \tag{4.29}$$

To upper bound the maximum number of transmissions with high probability, we note that Lemma 6 guarantees that

$$\max_{k=1,\ldots,T_{\text{ave}}} Q_k = O(\log T_{\text{ave}} + \log \epsilon^{-1}) \tag{4.30}$$

with high probability. Using Theorem 3, we can see that $O(\log T_{\text{ave}} + \log \epsilon^{-1}) =$

$O(\log n + \log \epsilon^{-1})$. Consequently, with probability greater than $1 - \epsilon/2$,

$$\mathcal{C}(n, \epsilon) = O\Big(\mathcal{E}(n, \epsilon)\big[\log n + \log \epsilon^{-1}\big]\Big) \ . \tag{4.31}$$

■

4.5 Simulations

Note that the averaging time is defined in equation (4.1) is a conservative measure, obtained by selecting the worst case initial field $x(0)$ for each algorithm. Due to this conservative choice, an algorithm is guaranteed to give (with high probability) an estimated average that is ϵ close to the true average for all choices of the underlying sensor observations. As we have theoretically demonstrated, our algorithm is provably superior to standard gossiping schemes in terms of this metric. In this section, we evaluate our geographic gossip algorithm experimentally on specific fields that are of practical interest. We construct three different fields and compare geographic gossip to the standard gossip algorithm with uniform neighbor selection probability. Note that for random geometric graphs, standard gossiping with uniform neighbor selection has the same scaling behavior as with optimal neighbor selection probabilities [Boyd *et al.*, 2005], which ensures that the comparison is fair.

Figures 4.2 through 4.4 illustrate how the cost of each algorithm behaves for various fields and network sizes. The error in the average estimation is measured by the normalized ℓ_2 norm $\frac{\|x(k) - x_{ave}\vec{1}\|}{\|x(0)\|}$. On the other axis we plot the total number of radio transmissions required to achieve the given accuracy. Figure 4.2 demonstrates how the estimation error behaves for a field that varies linearly. In Figure 4.3, we use a field that is created by placing temperature sources in the unit square and smooth the field by a simple process that models temperature diffusion. Finally, in Figure 4.4, we use a field that is zero everywhere except in a sharp spike in the center of the field. For this case, geographic gossip significantly outperforms standard gossip as the network size and time increase, except for large estimation tolerances ($\epsilon \approx 10^{-1}$) and small number of rounds.

As would be expected, simple gossip is capable of computing local averages quite fast. Therefore, when the field is sufficiently smooth, or when the averages in local node neighborhoods are close to the global average, simple gossip can generate approximate estimates that are closer to the true average with a smaller number of transmissions. For these cases, however, it is arguable that finding the global average is not of substantial interest in the first place. In all our simulations, the energy gains obtained by using geographic gossip were significant and asymptotically increasing for larger network sizes, corroborating our theoretical results.

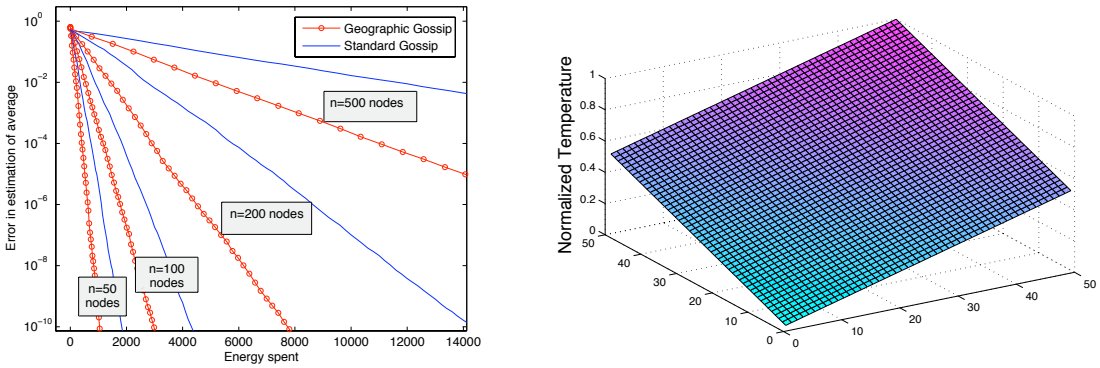


Figure 4.2: Estimation accuracy versus total spent energy for a linearly varying field.

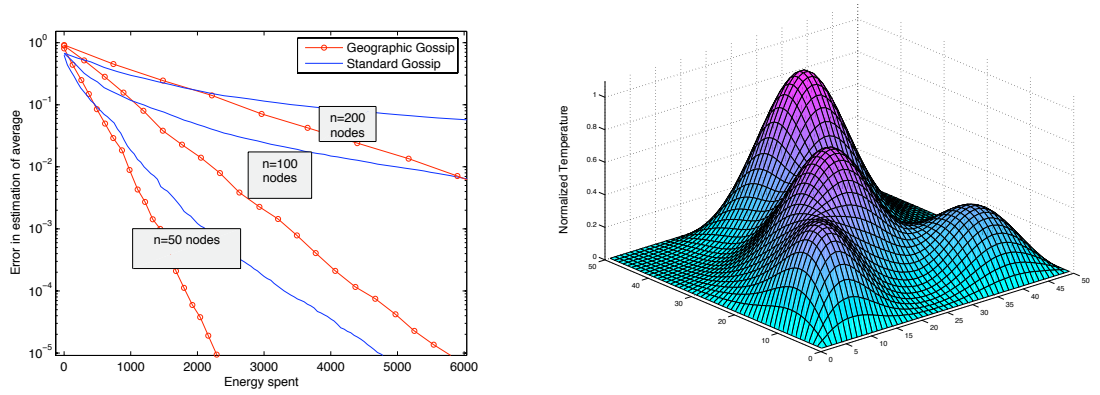


Figure 4.3: Estimation accuracy versus total spent energy for a smooth field modeling temperature.

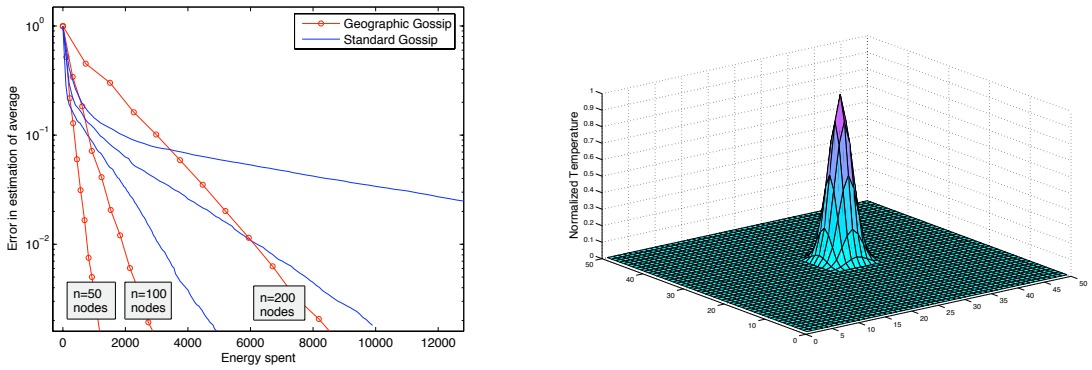


Figure 4.4: Estimation accuracy versus total spent energy for a field which is zero everywhere except in a sharp spike.

4.6 Discussion

We proposed and analyzed a novel message-passing algorithm for computing averages in networks in a distributed manner. By exploiting geographic knowledge of the network, our geographic gossip algorithm computes the averages faster than standard nearest-neighbor gossip. Even if the specific type of geographic routing considered here cannot be performed, similar gossip algorithms could be developed for any network structure that supports some form of routing to random nodes. Thus, our nearest-neighbor gossip can be understood as a particular case of a more general family of algorithms in which message-passing occurs on the overlay network supported by random routing. Other routing protocols may produce different overlay networks that could be analyzed in a similar manner.

We analyzed in detail the case of certain regular graphs, including the ring and grid networks, as well as the random geometric graph model, which is commonly used as a model of sensor networks under random deployments. Our algorithm can also be applied to other topologies that realistically model wireless sensor networks, and should provide gains when (a) the mixing time of a random walk on the graph is slow (b) efficient routing is possible, and (c) uniform sampling over space can yield approximately uniform sampling over sensors.

Although the current work has focused on the averaging problem, it is worth noting that many more complicated functions of interest can be computed using gossip; see the papers [Mosk-Aoyama and Shah, 2006; Spanos *et al.*, 2005; Rabbat *et al.*, 2005; Xiao *et al.*, 2005] for various examples involving localization, Kalman filtering and sensor fusion. However, linear operations (such as filtering) can be computed using our algorithm by allowing the sensors to pre-scale their observations

by their coefficients in the objective function. Our results suggest that geographic gossip may be useful instead of standard nearest-neighbor gossip to improve energy consumption in these and other distributed signal processing applications.

Chapter 5

Path Averaging

5.1 Introduction

In this chapter we investigate the performance of *path averaging*, which is the same algorithm as geographic gossip with the additional modification of *averaging all the nodes on the routed paths*. Observe that averaging the whole route comes almost for free in multihop communication, because a packet can accumulate the sum and the number of nodes visited, compute the average when it reaches its final destination and follow the same route backwards to disseminate the average to all the nodes along this route.

In path averaging, the selection of the routed path (and hence the routing algorithm) will affect the performance of the algorithm. We start by experimentally observing that the number of messages for grids and random geometric graphs seems to scale linearly when random greedy routing is used.

The mathematical analysis of path averaging with greedy routing is complex because the number of possible routes increases exponentially in the number of nodes. To make the analysis tractable we make two simplifications: a) We eliminate edge effects by assuming a grid or random geometric graph on a torus b) we use *box-greedy routing*, a scheme very similar to greedy routing with the extra restriction that each hop is guaranteed to be within a virtual box that is not too close or too far from the existing node. Box-greedy routing (described in section 5.3.4) can be implemented in a distributed way if each node knows its location, the location of its one-hope neighbors, and the total number of nodes n. We call path averaging with box-greedy routing *Box-path averaging*.

The main result of this chapter is that geographic gossip with path averaging requires $O(n)$ messages under these assumptions. Further, we present experimental evidence that suggests that this optimal behavior is preserved even when different routing algorithms are used.

	Grid	Random geometric graph
Standard gossip [Boyd *et al.*, 2006]	$\mathcal{E}(n,\epsilon) = \Theta(n^2 \log \epsilon^{-1})$	$\mathcal{E}(n,\epsilon) = \Theta\left(\frac{n^2 \log \epsilon^{-1}}{\log n}\right)$
Hops per time-slot	$\mathsf{E}[R] = \Theta(\sqrt{n})$	$\mathsf{E}[R] = \Theta\left(\sqrt{\frac{n}{\log n}}\right)$
Geographic gossip [Dimakis *et al.*, 2006c]	$T_{ave} = \Theta(n \log \epsilon^{-1})$	$T_{ave} = \Theta(n \log \epsilon^{-1})$
	$\mathcal{E}(n,\epsilon) = \Theta(n^{1.5} \log \epsilon^{-1})$	$\mathcal{E}(n,\epsilon) = \Theta\left(\frac{n^{1.5} \log \epsilon^{-1}}{\sqrt{\log n}}\right)$
Box-path averaging	$T_{ave} = \Theta(\sqrt{n} \log \epsilon^{-1})$	$T_{ave} = \Theta(\sqrt{n \log n} \log \epsilon^{-1})$
	$\mathcal{E}(n,\epsilon) = \Theta(n \log \epsilon^{-1})$	$\mathcal{E}(n,\epsilon) = \Theta(n \log \epsilon^{-1})$

Table 5.1: Performance of different Gossip algorithms. T_{ave} denotes ϵ-averaging time (in gossip rounds) and $\mathcal{E}(n,\epsilon)$ denotes expected number of messages required to estimate within ϵ accuracy.

5.2 Background and Metrics

5.2.1 Network model

Similarly to the previous chapter we model the wireless networks as random geometric graphs (RGG), following standard modeling assumptions [Gupta and Kumar, 2000; Penrose, 2003]. As analyzed in Chapter 4, to maintain connectivity and to minimize interference, the transmission radius $r(n)$ should scale like $r(n) = \sqrt{c \log n / n}$. For our analysis of path averaging, we prove a slightly stronger regularity condition: that in fact, if $\alpha > 2$, the number of nodes in each square will be $\Theta(\log n)$ nodes, i.e. the random geometric graphs are regular geometric graphs w.h.p. In Section 5.3.4, we assume that our network is a regular geometric graph embedded on a torus, and we ensure that any node in a square is able to communicate with any other node of its four neighboring squares by setting $c > 10$.

5.3 Path averaging algorithms

5.3.1 Path averaging on random geometric graphs.

The proposed algorithm combines gossip with random greedy geographic routing. A key assumption is that each node knows its location and is able to learn the geographic locations of its one-hop neighbors (for example using a single transmission per node). Also the nodes need to know the size of the space they are embedded in. Note that while our results are developped for random geometric topologies, the algorithm can be applied on any set of nodes embedded on some compact and convex region.

The algorithm operates as follows: at each time-slot one random node activates and selects a random position (target) on the unit square region where the nodes are spread out. Note that no node needs to be located on the target, since this would require global knowledge of locations. The node then creates a packet that contains its current estimate of the average, its position, the number of visited nodes so far (one), the target location, and passes the packet to a neighbor that is *randomly chosen among its neighbors closer to the target*. As nodes receive the packet, randomly and greedily forwarding it towards the target, they add their value to the sum and increase the hop counter. When the packet reaches its destination node (the first node whose nearest neighbors have larger distance to the target compared to it), the destination node computes the average of all the nodes on the path, and reroutes that information backwards on the same route. See Fig. 5.1 for an illustration of random greedy routing. It is not hard to show [Dimakis *et al.*, 2006c] that for $G(n, r)$ when r scales like $\Theta(\sqrt{\log n/n})$, greedy forwarding succeeds to reach the closest node to the random target with high probability over graphs — in other words there are no large 'holes' in the network. We will refer to this whole procedure of routing a message and averaging on a random path as one gossip round which lasts for one time-slot, after which $O(\sqrt{n/\log n})$ nodes will replace their estimates with their joint average. We prefer not to route the estimates by choosing the next node as the *closest* neighbor to the target, but as one random neighbor *closer* to the target, because we observed that the latter is cheaper (smaller C_c). Note that the nodes do not need to know the number of nodes n in the network, they only need the size of the field on which they are deployed.

5.3.2 Motivation–Performance simulations

We experimentally measured T_c and $\mathcal{C}_c$ in order to evaluate the performance of path averaging on random geometric graphs with a growing number n of nodes in the unit square. Fig 5.2(b) shows that our algorithm behaves strikingly better than standard gossip and geographic gossip, when, for example, $r(n) = \sqrt{c \log n/n}$ with

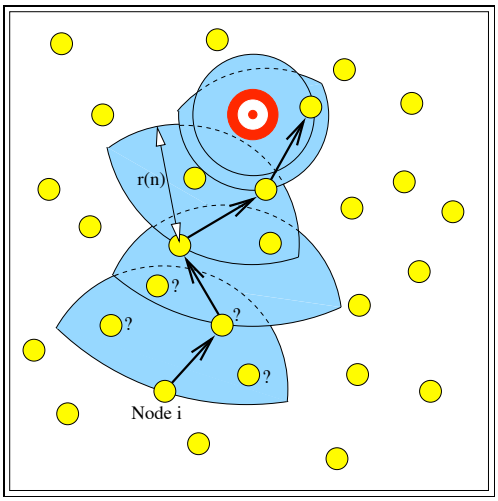


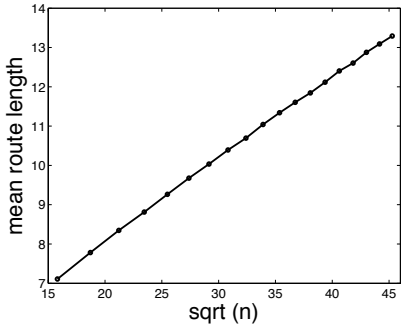
Figure 5.1: Random greedy routing. Node i has to choose the following node in the route among the nodes that are his neighbors (inside the ball of radius $r(n)$ centered in node i) *and* that are closer to the target than i (inside the ball of radius centered in the target, where d is the distance between node i and the target). Next node is thus randomly chosen in the intersection of the two balls.

$c = 4.5$. For other values of c, the performance of our algorithm also greatly improves previous gossip schemes. Most importantly, for small connection radius $r(n)$ (small c), the number of messages $\mathcal{C}_c$ behaves almost linearly in n (see Fig. 5.2(c)), and as c increases, the behavior improves (see Fig. 5.2(d)). The slight super-linearity in Fig.5.2(c) is due to small $r(n)$ and possibly edge effects. Clearly, we cannot expect better than linear behavior in n because at least n messages are necessary to average n values. Therefore path averaging with greedy routing seems to be optimal for sufficiently large constant c.

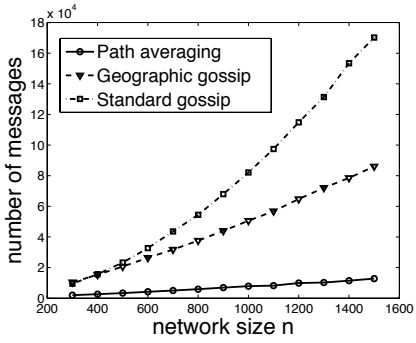
Unfortunately, the theoretical analysis of path averaging with greedy routing seems intractable. However, with a slight modification in the routing algorithm, and by ignoring edge effects, we are able to analyze path averaging, first for grids and then for regular geometric graphs. Recall that random geometric graphs are regular geometric graphs with high probability when n large if c is sufficiently large (Section 5.2.1).

5.3.3 $(\leftrightarrow, \updownarrow)$-path averaging on grids

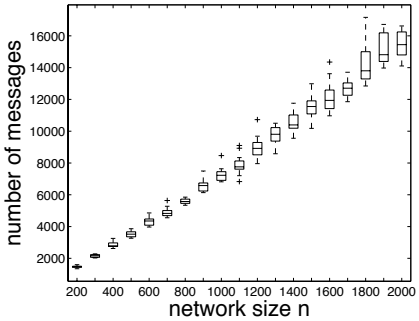
The first step in our analysis is understanding the behavior of path averaging on regular grids using a simple routing scheme. Throughout this chapter, a grid of n



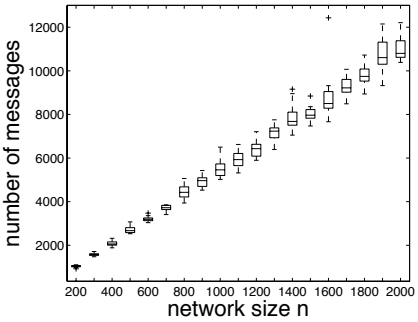
(a) Mean route length E(R).



(b) Consensus cost $\mathcal{C}_c$: compare three methods

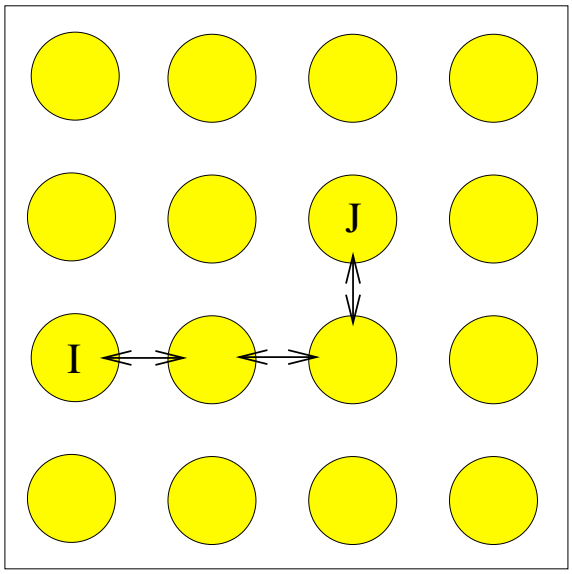


(c) $\mathcal{C}_c$: path averaging, $r(n) = \sqrt{4.5 \log n/n}$



(d) $\mathcal{C}_c$: path averaging, $r(n) = \sqrt{25 \log n/n}$

Figure 5.2: Performance of path averaging. The simulations were performed over 15 graphs per n. Averaging time was measured here by $T_c \simeq (t_1 - t_2)/[\log \|\epsilon(t_2)\| - \log \|\epsilon(t_1)\|]$ for $t_1 = 500$ and $t_2 = 1750$. (a) The mean route length in random greedy routing behaves in $\sqrt{n/\log n}$. (b) Comparison between standard gossip, geographic gossip (without rejection sampling) and path averaging with $r(n) = \sqrt{4.5 \log n/n}$. (c), (d) Consensus costs $\mathcal{C}_c = \mathrm{E}[R]T_c$ for radii $r(n) = \sqrt{4.5 \log n/n}$ and $r(n) = \sqrt{25 \log n/n}$.



(a) $(\leftrightarrow, \updownarrow)$-route

Figure 5.3: (a) Shortest $(\leftrightarrow, \updownarrow)$-route from I to J on the grid. (b) Example of box-path averaging on an RGG: The node with inital value 3 selects a random position and places a target. Using $(\leftrightarrow, \updownarrow)$-box routing towards that target, all the nodes on the path replace their values with the average of the four nodes.

nodes will be a 4-connected lattice on a torus of size $\sqrt{n} \times \sqrt{n}$. $(\leftrightarrow, \updownarrow)$-path averaging performs as follows: At each iteration t, a randomly selected node I wakes up and selects a random destination node J so that the pair (I, J) is independently and uniformly distributed. Node I also flips a fair coin to design the first direction: horizontal $(\leftrightarrow)$ or vertical $(\updownarrow)$. If for instance horizontal was picked as the first direction, the path between I and J is then defined by the shortest horizontal-vertical route between I and J (see Fig. 5.3(a)). The estimates of all the nodes on this path are aggregated and averaged by messages passed on this path, and at the end of the iteration the estimates of the nodes on this path are updated to their global average. Clearly, this message-passing procedure can be executed if each node knows its location on the grid.

5.3.4 Box-path averaging on regular geometric graphs

As seen in Section 5.2.1, a regular geometric graph can be organized in virtual squares with the transmission radius $r(n)$ selected so that a node can pass messages to any node in the four squares adjacent to its own square.

In box-path averaging, when a node activates, it chooses uniformly at random a target location in the unit torus and its initial direction: horizontal or vertical. Then a node is selected uniformly from the ones in the adjacent square in the right direction. (Recall that regularity ensures that w.h.p. $\Theta(\log n)$ nodes will be in each square.) The routing stops when the message reaches a node in the square where the target is located. As in the previous path averaging algorithms, the estimates of all the nodes on the path are averaged and all the nodes replace their values with this estimate (see Fig. 5.4). The key point is that box-path averaging can be executed if each node knows its location, the locations of its one-hop neighbors and the total number of nodes n, because with this knowledge each node can figure out which square it belongs to and pass messages appropriately.

Box-greedy routing is a regularized version of random greedy routing, and is introduced to make the analysis tractable. Both routing schemes proceed by choosing the next hop among $\Theta(\log n)$ nodes (Fig. 5.5). Box-greedy routing generates routes with $\Theta(\sqrt{n/\log n})$ hops on average, and random greedy routing does as well on experiments (Fig. 5.2(a)). We are now ready to start the theoretical analysis of the aforementioned path averaging algorithms.

5.4 Analysis

5.4.1 Averaging and eigenvalues.

Let $x(t)$ denote the vector of estimates of the global averages after the t^{th} gossip round, where $x(0)$ is the vector of initial measurements. Any gossip algorithm can be described by an equation of the form

$$x(t+1) = W(t)x(t), \tag{5.1}$$

where $W(t)$ is the averaging matrix over the t^{th} time-slot.

We say that the algorithm converges almost surely (a.s.) if $P[\lim_{t\to\infty} x(t) = x_{ave}\vec{1}] = 1$. It converges in expectation if $\lim_{t\to\infty} \mathsf{E}[x(t) - x_{ave}\vec{1}] = 0$, and there is mean square convergence if $\lim_{t\to\infty} \mathsf{E}[\|x(t) - x_{ave}\vec{1}\|_2] = 0$. There are two *necessary* conditions for convergence:

$$\begin{cases} \vec{1}^T W(t) = \vec{1}^T \\ W(t)\vec{1} = \vec{1}, \end{cases} \tag{5.2}$$

which respectively ensure that the average is preserved at every iteration, and that $\vec{1}$ is a fixed point. For any linear distributed averaging algorithm following (5.1) where $\{W(t)\}_{t\geq 0}$ is i.i.d., conditions for convergence in expectation and in mean square can be found in [Boyd *et al.*, 2004]. In gossip algorithms, $W(t)$ are symmetric and projec-

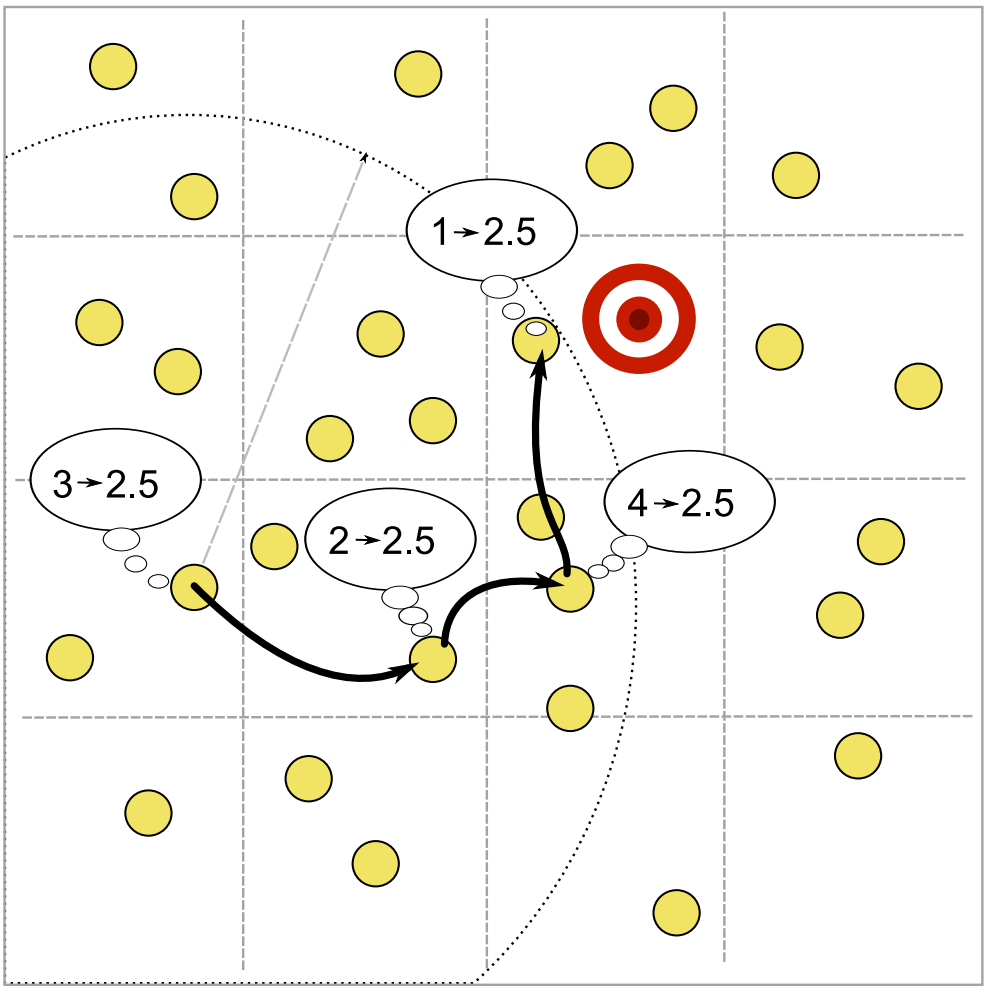


Figure 5.4: Example of box-path averaging: The node with inital value 3 selects a random position and places a target. Using $(\leftrightarrow, \updownarrow)$-box routing towards that target, all the nodes on the path replace their values with the average of the four nodes.

tion matrices. Taking into account this particularity, we can state specific conditions for convergence. Let $\lambda_2(\mathsf{E}[W])$ be the second largest eigenvalue in magnitude of the expectation of the averaging matrix $\mathsf{E}[W] = \mathsf{E}[W(t)]$. If condition (5.2) holds and if $\lambda_2(\mathsf{E}[W]) < 1$, then $x(t)$ converges to $x_{ave}\vec{1}$ in expectation and in mean square.

In the case where $\{W(t)\}_{t\geq 0}$ is stationary and ergodic (and thus in particular when $\{W(t)\}_{t\geq 0}$ is i.i.d.), sufficient conditions for a.s. convergence can be proven [Denantes, 2007]: if the gossip communication network is connected, then the estimates of gossip converge to the global average $\bar{x}_{\text{ave}}$ with probability 1. More precisely, define $T_\eta :=$

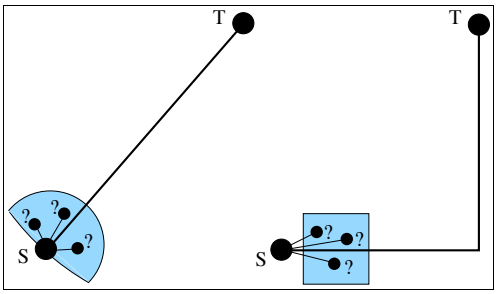


Figure 5.5: Choosing next node in the route. On the left: random greedy routing, on the right: $(\updownarrow, \leftrightarrow)$-box routing. It is easy to see that the two choice areas contain on average $\Theta(\log n)$ nodes.

$\inf\{t \geq 1 : \prod_{p=0}^{t} W(t-p) \geq \eta > 0\}$. T_η is a stopping time. If $\mathsf{E}[T_\eta] < \infty$, then the estimates converge to the global average with probability 1. In other words, every node has to eventually connect to the network, which has to be jointly connected.

Interestingly, the value of $\lambda_2(\mathsf{E}[W])$, that appears in the criteria of convergence in expectation and of mean square convergence, controls the speed of convergence:

$$T_c(\mathsf{E}[W]) \leqslant \frac{2}{\log\left(\frac{1}{\lambda_2(\mathsf{E}[W])}\right)} \leq \frac{2}{1-\lambda_2(\mathsf{E}[W])}. \tag{5.3}$$

A straightforward extension of the proof of Boyd et al. [Boyd *et al.*, 2006] from the case of pairwise averaging matrices to the case of symmetric projection averaging matrices yields the following bound on the ϵ-averaging time, which also involves $\lambda_2(\mathsf{E}[W])$:

$$T_{ave}(\epsilon, \mathsf{E}[W]) \leq \frac{3\log\epsilon^{-1}}{\log\left(\frac{1}{\lambda_2(\mathsf{E}[W])}\right)} \leq \frac{3\log\epsilon^{-1}}{1-\lambda_2(\mathsf{E}[W])}. \tag{5.4}$$

There is also a lower bound of the same order, which implies that $T_{ave}(\epsilon, \mathsf{E}[W]) = \Theta(\log\epsilon^{-1}/(1-\lambda_2(\mathsf{E}[W])))$.

Consequently, the rate at which the *spectral gap* $1-\lambda_2(\mathsf{E}[W])$ approaches zero as n increases, controls both the ϵ-averaging time T_{ave} and the consensus time T_c. For example, in the case of a complete graph and uniform pairwise gossiping, one can show that $\lambda_2(\mathsf{E}[W]) = 1 - 1/n$. Therefore, as previously mentioned, the consensus time of this scheme is $O(n)$. In pairwise gossiping, the convergence time and the number of messages have the same order because there is a constant number R of transmissions per time-slot. In geographic gossip and in path averaging on random geometric graphs, one round uses many messages for the path routing ($\sqrt{n/\log n}$ messages on

average), hence multiplying the order of consensus time $T_c(n)$ by $\sqrt{n/\log n}$ gives the order of consensus cost $\mathcal{C}_c(n)$.

5.4.2 The travel agency method

A direct consequence of the previous section is that the evaluation of consensus time requires an accurate upper bound on $\lambda_2(\mathsf{E}[W])$. Consequently, computing the averaging time of a scheme takes two steps: (1) evaluation of $\mathsf{E}[W]$, (2) upperbound of its second largest eigenvalue in magnitude. $\mathsf{E}[W]$ is a doubly stochastic matrix that corresponds to a time-reversible Markov Chain.

We can therefore use techniques developed for bounding the spectral gap of Markov Chains to bound the convergence time of gossip. In particular, we will use Poincaré's inequality by Diaconis and Stroock [Diaconis and Stroock, 1991b] (see also [Brémaud, 1999], p. 212-213 and the related canonical paths technique [Sinclair, 1992]) to develop a bounding technique for gossip.

Theorem 4 (Poincaré's inequality [Diaconis and Stroock, 1991b])**.** *Let P denote an $n \times n$ irreducible and reversible stochastic matrix, and π its left eigenvector associated to the eigenvalue 1 ($\pi^T P = \pi^T$) such that $\sum_{i=1}^{n} \pi(i) = 1$. A pair $e = (k, l)$ is called an edge if $P_{kl} \neq 0$. For each ordered pair (i, j) where $1 \leqslant i, j \leqslant n$, $i \neq j$, choose one and only one path $\gamma_{ij} = (i, i_1, \ldots, i_m, j)$ between i and j such that (i, i_1), (i_1, i_2), $\ldots$, (i_m, j) are all edges. Define*

$$|\gamma_{ij}| = \frac{1}{\pi(i)P_{ii_1}} + \frac{1}{\pi(i_1)P_{i_1 i_2}} + \ldots + \frac{1}{\pi(i_m)P_{i_m j}}. \tag{5.5}$$

The Poincaré coefficient is defined as

$$\kappa = \max_{edge\ e} \sum_{\gamma_{ij} \ni e} |\gamma_{ij}| \pi(i) \pi(j). \tag{5.6}$$

Then the second largest eigenvalue of P verifies

$$\lambda_2(P) \leq 1 - \frac{1}{\kappa}. \tag{5.7}$$

We will apply this theorem with $P = \mathsf{E}[W]$. Here $\pi(i) = 1/n$ for all $1 \leqslant i \leqslant n$.

The combination of Poincaré inequality with bounds 5.3 and 5.4 forms a versatile technique for bounding the performance of gossip algorithms that we call the *travel agency* method. It is crucial to understand that the edges used in the application

of the theorem are abstract and do not correspond to actual edges in the physical network. They instead correspond to paths on which there is joint averaging, and hence information flow, through message-passing. Consider the following analogy. Imagine that n airports are positioned at the locations of the nodes of the network. In this scenario, we are given a table $P = \mathsf{E}[W]$ of the flight capacities (number of passengers per time unit) between any pair of airports among the n airports. A good *averaging intensity* $\mathsf{E}[W_{ij}]$ between nodes i and j correspond to a good *capacity* flight between airports i and j in the travel agency method. Here edges e are existing flights and, in our specific case, there is the same number of travelers in all the airports ($\pi(i) = 1/n$ for all i). We are asked to design one and only one road map γ_{ij} between each pair of airports i and j that avoids congestion and multiple hops. $|\gamma_{ij}|$ measures the level of congestion between airport i and airport j. The theorem tells us that if we can come up with a road map that avoids significant congestion on the worst flight (i.e. if κ is small), then we will have proven that the flying network is efficient (λ_2 is small). The previous bounds 5.3,5.4 can now be used to bound the consensus time and consensus cost.

One of the important benefits of this bounding technique is that we do not need know the entries of $\mathsf{E}[W]$ to bound the averaging cost, and only good lower bounds suffice. In terms of the analogy, we only need to know that each flight (i, j) has at least capacity $C_{i,j}$. If (i, j) can actually carry more passengers ($P_{i,j} \geqslant C_{i,j}$), then our measure of congestion κ will be overestimated. While our final upper-bounds will not be as tight as they could have been if we had exact knowledge of $\mathsf{E}[W]$, they suffice to establish the optimal asymptotic behavior.

5.4.3 Example: standard gossip revisited

In order to illustrate the generality of our technique, we show how to apply it on simple examples, by giving sketches of novel proofs for known results on nearest neighbors gossip on the complete graph and on the random geometric graph.

5.4.3.1 Complete graph

For any $i \neq j$, $\mathsf{E}[W_{ij}] = 1/n^2$. Indeed $W_{ij} = 0.5$ when node i wakes up (event of probability $1/n$) and chooses node j (event of probability $1/n$ as well), or when j wakes up and chooses i. We apply now the travel agency method. We see in $\mathsf{E}[W]$ that all flights have equal capacity $1/n^2$ and that there are direct flights between any pair of airports. We choose here the simplest road map one could think of: to go from airport i to airport j, each traveller should take the direct hop $\gamma_{ij} = (i, j)$. Then the sum in (5.5) has only one term: $|\gamma_{ij}| = n^3$. In this case all flights are equal and one flight $e = (i, j)$ belongs only to one road map: γ_{ij}. Thus the sum in (5.6) also

has only one term and $\kappa = n^3/(n \cdot n) = n$. Therefore $\lambda_2(\mathsf{E}[W]) \leqslant 1 - 1/n$, which proves that $T_c(n) = O(n)$. Note that the complete graph is the overlay network of geographic gossip [1] (every pair of node can be averaged at the expense of routing), which thus performs in $\mathcal{C}_c(n) = O(n\sqrt{n/\log n})$.

5.4.3.2 Random geometric graph (RGG)

In Section 5.5.3 subsequently show that if the connection radius $r(n)$ is large enough, then RGGs are regular with high probability, i.e. each node has $\Theta(\log n)$ neighbors. To keep the illustration of the travel agency method simple, we assume that the nodes lie on a torus (no border effects). Consider the pair of nodes (i, j). If i and j are not neighbors, then $\mathsf{E}[W_{ij}] = 0$; if i and j are neighbors, then $\mathsf{E}[W_{ij}] = \Theta\left(1/(n \log n)\right)$ because node i wakes up with probability $1/n$ and chooses node j with probability $\Theta\left(1/\log n\right)$. We now have to create a roadmap with only short distance paths. Regularity ensures that there are no isolated nodes that could create local congestion. We thus naturally decide that the best way to go is to select paths along the straightest possible line between the departure airport and the destination airport. This will require $O(\sqrt{n/\log n})$ hops, therefore the right hand side of Equation (5.5) is the sum of $O(\sqrt{n/\log n})$ terms, each of equal order:

$$|\gamma_{ij}| = O\left(\sqrt{\frac{n}{\log n}}\right) \frac{1}{1/n} \Theta\left(\frac{1}{1/n \log n}\right) = O(n^2\sqrt{n \log n}). \tag{5.8}$$

Now we need to compute in how many paths each particular flight is used. It follows from our regularity and torus assumptions that each flight appears in approximatively the same number of road maps. There are n^2 paths that use $O(\sqrt{n/\log n})$ flights, but there are only $\Theta(n \log n)$ different flights, hence each flight is used in $O\left((n/\log n)^{1.5}\right)$ paths. We can now compute the Poincaré coefficient κ. We drop the $\max_e$ argument in Equation (5.6) because all flights are equal. As $\pi(i) = \pi(j) = 1/n$,

$$\kappa = \sum_{\gamma_{ij} \ni e} O(n^2\sqrt{n \log n}) \frac{1}{n} \frac{1}{n} \tag{5.9}$$

$$= O\left((\frac{n}{\log n})^{1.5}\right) O(\sqrt{n \log n}) \tag{5.10}$$

$$= O(\frac{n^2}{\log n}), \tag{5.11}$$

which proves that $T_c(n) = O(n^2/\log n)$.

[1] In reality, geographic gossip will not be completely uniform but rejection sampling can be used [Dimakis *et al.*, 2006c] to tamper the distribution

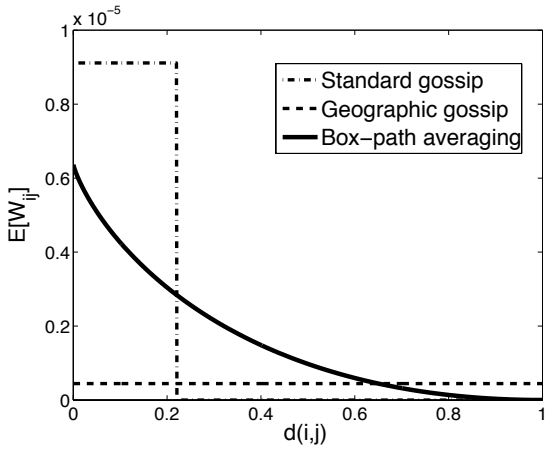


Figure 5.6: Behavior of $\mathsf{E}[W_{ij}]$ as a function of the distance in norm 1 between i and j for standard gossip, geographic gossip and box-path averaging.

5.4.3.3 Comments

The proof of the performance of path averaging on a RGG given in Section 5.5.2 gives insight on how to complete this last proof. It is interesting to see that the travel agency method describes how information will *diffuse* in the network. In the second example, far away nodes will never directly average their estimates, but they will do it indirectly, using the nodes between them.

Note that our method does not give lower-bounds on $\lambda_2(\mathsf{E}[W])$, which would be useful to give an equivalent order for ϵ-averaging time T_{ave}. In the case of path averaging, this is not an issue since it is not possible to achieve better than the consensus cost $\mathcal{C}_c(n) = \Theta(n)$. So if the method shows that $T_c(n) = O(\sqrt{n \log n})$, we have that $\mathcal{C}_c(n) = O(\sqrt{n \log n})O(\sqrt{n/\log n}) = O(n)$ and we can conclude that $\mathcal{C}_c(n) = \Theta(n)$.

5.4.4 Main Results

The main results of this chapter is that the consensus cost of $(\leftrightarrow, \updownarrow)$-path averaging on grids and of box-path averaging on random geometric graphs, behave *linearly* in the number of nodes n:

Theorem 5 ($(\leftrightarrow, \updownarrow)$-path averaging on grids)**.** *On a $\sqrt{n} \times \sqrt{n}$ torus grid, the consensus time $T_c(n)$ of $(\leftrightarrow, \updownarrow)$-path averaging, described in Section 5.3.3, is $O(\sqrt{n})$. Furthermore, the consensus cost is linear: $\mathcal{C}_c(n) = O(n)$.*

Theorem 6 (Box-path averaging on RGG)**.** *Consider a random geometric graph* $G(n,r)$ *on the unit torus with* $r(n) = \sqrt{\frac{c \log n}{n}}$, $c > 10$. *With high probability over graphs, the consensus time* $T_c(n)$ *of box-path averaging, described in Section 5.3.4, is* $O(\sqrt{n \log n})$. *Furthermore, the consensus cost is linear:* $\mathcal{C}_c(n) = O(n)$.

The proofs of Theorem 5 and Theorem 6 are given in the Appendix. Both proofs have the same structure: we first lower bound the entries of $\mathsf{E}[W]$ and next upper bound its second largest eigenvalue in magnitude. Figure 5.6 shows the behavior of $\mathsf{E}[W_{ij}]$ as a function of the L_1 distance between nodes i and j for standard gossip, geographic gossip and path averaging; respectively the proofs give us the insight behind the good performance of box-path averaging compared to standard gossip and geographic gossip by simply analysing Fig. 5.6. Box-path averaging concentrates the *averaging intensities* $\mathsf{E}[W_{ij}]$ of node i in the area of nodes j close to i. Indeed, the closer two nodes, the higher the probability that they are on the same route. Thus, as we can observe on Fig. 5.6, close nodes have a much higher averaging intensity $\mathsf{E}[W_{ij}]$ than in geographic gossip, where nodes are equally rarely averaged together (the proof shows an order $\sqrt{n/\log n}$ higher). However, the averaging intensity gained by close nodes is lost for far away nodes, which do not average together well anymore (a factor n loss compared to geographic gossip).

In terms of the travel agency method, in box-path averaging over the unit area torus, flights with that cover distances shorter than $1/2$ have high capacity, whereas long distance flights are rare. To apply the method, the idea is to chose 2-hop paths: to go from node i to node j, the path will contain two hops that stop half way, in order to exclusively and fairly use the high capacity flights. Remember that standard gossip needs $\sqrt{n/\log n}$ flights per path (see Section 5.4.3.1), which heavily penalizes the performance despite a very high averaging intensity $\mathsf{E}[W_{ij}]$ for neighboring nodes i and j (see Fig. 5.6, where $\mathsf{E}[W_{ij}]$ is large for neighboring nodes but falls to 0 for distances larger than $r(n)$). The performance of path averaging algorithms is good thanks to a diffusion scheme requiring only $O(1)$ flights in each path and $O(1)$ uses of each flight in the road map, combined with a high enough level of averaging intensity $\mathsf{E}[W_{ij}]$. Each node can act as a diffusion relay for some far away nodes, so that the whole network can benefit from the concentration of the averaging intensity.

As a summary, in contrast with geographic gossip, path averaging and standard gossip *concentrate* their averaging intensity on close nodes, which leads to larger coefficients $\mathsf{E}[W_{i,j}]$ when nodes i and j are close enough. However, while standard gossip pays for its concentration with long paths overusing every existing flight, the diffusion pattern of path averaging operates in 2 steps only without creating any congestion (more precisely, we compute in the proof that each flight is used in at most 9 paths). In conclusion, the analysis shows that path averaging achieves a good tradeoff between promoting *local* averaging to increase averaging intensity (large $\mathsf{E}[W_{ij}]$) and

favoring *long distance* averaging to get an efficient diffusion pattern (every path γ_{ij} contains only $O(1)$ edges, and every edge e appears in only $O(1)$ paths).

5.5 Appendix

We start with a reminder of notation and some definitions.

- $G(n, r)$ or RGG: random geometric graph with n nodes and connection radius r.
- $x(0)$: vector of the initial values to be averaged.
- $\bar{x}_{\text{ave}} = \sum_{k=1}^{n} x_k(0)/n$.
- $x(t)$: vector of the estimates of the average.
- $S(t)$: the random set of nodes that average together at time-slot t.
- $R(t)$: number of one hop transmissions at time-slot t.
- $\epsilon(t) = x(t) - \bar{x}_{\text{ave}}\vec{1}$: error vector, where $\vec{1}$ is the vector of all ones.
- $W(t)$: averaging matrix at time t.
- λ_2: second largest eigenvalue in magnitude.
- γ_{ij}: path starting in i and ending in j.
- $|\gamma_{ij}|$ measures the "resistance" of path γ_{ij} (Eq. (5.5)).
- κ: Poincaré coefficient (Eq. (5.6)).
- $T_{ave}(\epsilon)$: ϵ-averaging time (Def. 4.1)
- $C_{ave}(\epsilon) = \mathsf{E}[R(1)]T_{ave}$: expected ϵ-averaging cost.
- T_c, C_c: consensus time, consensus cost (Def. 4.5, 2).

5.5.0.1 List of the algorithms

- Standard gossip: pairwise gossip where only direct neighbors can average their estimates together.
- Geographic gossip: pairwise gossip where any pair of nodes can average their estimates together at the expense of routing.
- Path averaging: at each iteration a random route is created by random greedy routing in an RGG. The nodes of the route average their estimates together.
- $(\leftrightarrow, \updownarrow)$-path averaging: at each iteration a random route is created by $(\leftrightarrow, \updownarrow)$-routing on a grid (embedded on a torus in the analysis). The nodes of the route average their estimates together.
- Box-path routing: at each iteration a random route is created by box-routing on a regular geometric graph (embedded on a torus in the analysis). The nodes of the route average their estimates together.

5.5.1 Performance of $(\leftrightarrow, \updownarrow)$-path averaging on a grid

This section prooves Theorem 5, which states the linearity of consensus cost for $(\leftrightarrow, \updownarrow)$-path averaging on a grid. The analyzed algorithm is described in Section 5.3.3.

We need to define the shortest distance on a torus. To this end, we introduce a torus absolute value $|.|_{\mathcal{T}}$ and a torus L_1 norm $\|.\|_1$. For any algebraic value x on a one dimensional torus (circle with $\sqrt{n}$ nodes) and any vector i on a $\sqrt{n} \times \sqrt{n}$ two dimensional torus,

$$\begin{aligned} |x|_{\mathcal{T}} &= \min(|x|, |x - \sqrt{n}|, |x + \sqrt{n}|) \\ \|i\|_1 &= |i_x|_{\mathcal{T}} + |i_y|_{\mathcal{T}}. \end{aligned}$$

We call $\ell_{ij} = \|j - i\|_1$ the L_1 distance between nodes i and j. The shortest routes between I and J have $\alpha = \ell_{IJ} + 1 = |J_x - I_x|_{\mathcal{T}} + |J_y - I_y|_{\mathcal{T}} + 1$ nodes to be averaged, thus the non-zero coefficients of their corresponding matrices W are all equal to $1/\alpha$.

To each route r, we assign a generalized gossip $n \times n$ matrix $W^{(r)}$ that averages the current estimates of the nodes on the route. Consequently, at iteration t, $W(t) = W^{(r(t))}$, where $r(t)$ was randomly chosen. We call R the route random variable, $s(R)$ its starting node, $d(R)$ its destination node, and $\ell(R) = \ell_{s(R)d(R)} + 1$ its number of nodes. As we choose the shortest route, the maximum number of nodes a route can contain is $\sqrt{n}$ if $\sqrt{n}$ is odd, $\sqrt{n} + 1$ if $\sqrt{n}$ is even, which can be written as

$2\lfloor\sqrt{n}/2\rfloor + 1$ in short.

5.5.1.1 Evaluating $\mathsf{E}[W]$

Lemma 7. (Expected $\mathsf{E}[W]$ on the grid) *For any pair of nodes (i,j), if their distance normalized to the maximum distance $\delta_{ij} = \|j-i\|_1/\sqrt{n}$ is smaller than a constant, then*

$$\mathsf{E}[W_{i,j}] = \Omega\left(\frac{1}{n^{1.5}}\right). \tag{5.12}$$

More precisely,

$$\mathsf{E}[W_{i,j}] \geq \frac{2(1-\delta_{ij}+\delta_{ij}\log\delta_{ij})}{n\sqrt{n}}.$$

Therefore, as expected, far away nodes are less likely to be jointly averaged compared to neighboring ones (see Figure 5.6).

Proof. Observing that $\mathbb{E}[W^{(R)}|(\leftrightarrow,\updownarrow)] = \mathbb{E}[W^{(R)}|(\updownarrow,\leftrightarrow)]$ because the route from a node I to a node J horizontally first has the same nodes as the route from J to I vertically first, we get

$$\begin{aligned}\mathsf{E}[W] &= \mathbb{E}[W^{(R)}] \\ &= \frac{1}{2}\mathbb{E}[W^{(R)}|(\leftrightarrow,\updownarrow)] + \frac{1}{2}\mathbb{E}[W^{(R)}|(\updownarrow,\leftrightarrow)] \\ &= \mathbb{E}[W^{(R)}|(\leftrightarrow,\updownarrow)].\end{aligned}$$

So, for a given pair of nodes (i,j), we can compute the (i,j)th entry of the matrix expectation $\mathsf{E}[W]$ by systematically routing first horizontally. Only the $(\leftrightarrow,\updownarrow)$-routes which contain both these two nodes i and j will have a non-zero contribution in $\mathsf{E}[W_{ij}]$. Pick such a route r, the (i,j)th entry of the corresponding averaging matrix is $W_{i,j}^{(r)} = 1/\ell(r)$. We call $\mathcal{R}_{ij}^{\ell}$ the set of $(\leftrightarrow,\updownarrow)$-routes with ℓ nodes passing by node i and by node j, and denote $x^+ = \max(x,0)$. It is not hard to see that $(\ell-\ell_{ij})^+$ is the number of routes of length ℓ passing by i first and j next (see Fig. 5.7), so

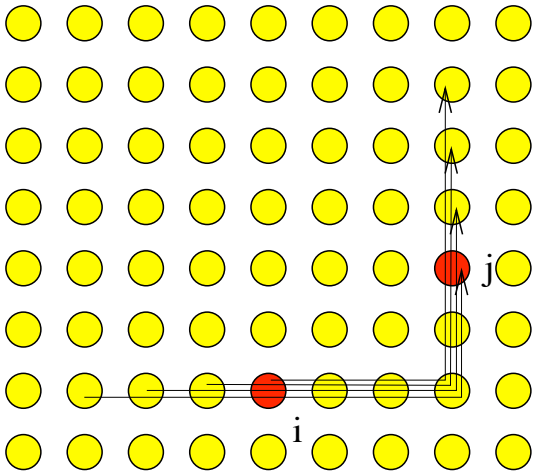


Figure 5.7: Counting the number of routes of length $\ell = 9$ nodes, in the case where $\ell_{ij} = 5$. There are $\ell - \ell_{ij} = 9 - 5 = 4$ possible routes with exactly ℓ nodes going through node i then through node j. We admit only routes going horizontally first then vertically.

$|\mathcal{R}_{ij}^{\ell}| = 2(\ell - \ell_{ij})^{+}$. We thus have for any $i \neq j$:

$$\begin{aligned}
\mathsf{E}[W_{i,j}] &= \sum_r W_{i,j}^{(r)} \mathbb{P}[R = r] \\
&= \frac{1}{n^2} \sum_r W_{i,j}^{(r)} \\
&= \frac{1}{n^2} \sum_{\ell=\ell_{ij}+1}^{2\lfloor \frac{\sqrt{n}}{2} \rfloor + 1} \frac{|\mathcal{R}_{ij}^{\ell}|}{\ell} \\
&= \frac{2}{n^2} \sum_{\ell=\ell_{ij}+1}^{2\lfloor \frac{\sqrt{n}}{2} \rfloor + 1} \frac{\ell - \ell_{ij}}{\ell},
\end{aligned}$$

from which we can deduce that for $i \neq j$

$$\begin{aligned}
\mathsf{E}[W_{i,j}] &\leq \frac{2}{n^2}\int_{\ell_{ij}+1}^{\sqrt{n}+2} \frac{x-\ell_{ij}}{x}dx \\
&= \frac{2}{n^2}\left(\sqrt{n}-\ell_{ij}+1-\ell_{ij}\ln\frac{\sqrt{n}+2}{\ell_{ij}+1}\right) \\
\mathsf{E}[W_{i,j}] &\geq \frac{2}{n^2}\int_{\ell_{ij}}^{\sqrt{n}} \frac{x-\ell_{ij}}{x}dx \\
&= \frac{2}{n^2}\left(\sqrt{n}-\ell_{ij}-\ell_{ij}\ln\frac{\sqrt{n}}{\ell_{ij}}\right).
\end{aligned}$$

$\mathsf{E}[W_{i,j}]$ decreases from $\frac{2}{n\sqrt{n}}$ to $o(\frac{1}{n^2})$ as a function of ℓ_{ij}. To get a normalized expression with respect to $\sqrt{n}$, we use the coefficient δ_{ij} defined in the statement of Lemma 7.

$$\begin{aligned}
&\frac{2}{n\sqrt{n}}\left(1-\delta_{ij}+\delta_{ij}\ln\delta_{ij}\right) \leq \mathsf{E}[W_{i,j}] \leq \\
&\quad \frac{2}{n\sqrt{n}}\left(1-\delta_{ij}+\delta_{ij}\ln\delta_{ij}+\frac{1}{\sqrt{n}}-\delta_{ij}\ln\frac{\sqrt{n}+2}{\sqrt{n}+\frac{1}{\delta_{ij}}}\right).
\end{aligned}$$

This establishes the claim. In particular, if $\delta_{ij} = 1/2$, then $\mathsf{E}[W_{i,j}] \sim \frac{1-\ln 2}{n\sqrt{n}}$.
■

5.5.1.2 Bounding $\lambda_2(\mathsf{E}[W])$

We need now to upperbound the second largest eigenvalue in magnitude of $\mathsf{E}[W]$, or equivalently, the relaxation time $1/(1-\lambda_2(\mathsf{E}[W]))$.

Lemma 8 (Relaxation time)**.**

$$\frac{1}{1-\lambda_2(\mathsf{E}[W])} = O(\sqrt{n}). \tag{5.13}$$

Proof. The Poincaré inequality (Theorem 4) bounds the second largest eigenvalue of a stochastic matrix and not necessarily its second largest eigenvalue *in magnitude*, which is the important quantity involved in Eq. (5.3). It could happen that the smallest negative eigenvalue is larger in magnitude than the second largest eigenvalue. Consequently, if we show that all the eigenvalues of $\mathsf{E}[W]$ are positive, then the two eigenvalues coincide and we can use the Poincaré inequality to bound the second largest eigenvalue in magnitude. $\mathsf{E}[W]$ is symmetric so all its eigenvalues are real. The sum of all the entries along the lines of $\mathsf{E}[W]$ without counting the diagonal element is

$O(1/\sqrt{n})$, whereas the diagonal elements are $\Theta(1)$, so by Gershgorin bound [Brémaud, 1999], all the eigenvalues of $\mathsf{E}[W]$ are positive.

We can now use the bounds on $\mathsf{E}[W]$ to bound its spectral gap.

We want to prove that path averaging performs $\sqrt{n}$ better than geographic gossip, where $\mathsf{E}[W_{i,j}] = 1/n^2$ (5.4.3.1). It is encouraging to note that for $\delta_{ij} \leqslant 1/2$, $\mathsf{E}[W_{i,j}] \geqslant \frac{1-\ln 2}{n\sqrt{n}}$, which is precisely $\sqrt{n}$ better than $1/n^2$. We thus observe that it is possible to find edges with a good capacity with length equal to half of the whole graph. However very distant destinations remain problematic. Consider the extreme case of a distance $\sqrt{n}$ between two nodes i and j. There are only two routes that will jointly average them: the route that goes from i to j, and the reverse one. These routes are selected with probability $1/n^2$ and $W_{ij} = 1/\sqrt{n}$, implying that $\mathsf{E}[W_{ij}] = 2/n^{2.5} \ll 1/n^{1.5}$.

Formally, for each ordered and distinct pair (i, j), we choose a 2-hop path γ_{ij} from i to j stopping by an "airport" node k chosen to be located approximatively half way between i and j. To be more precise, we define direction functions σ_x and σ_y, where $\sigma_x(i, j) = 1$ (respectively, $\sigma_y(i, j) = 1$) if the horizontal (resp., vertical) part of the route from i to j goes to the right (resp., up) and $\sigma_x(i, j) = -1$ (resp., $\sigma_y(i, j) = -1$) if it goes left (resp., down). The coordinates of k in the torus are:

$$\begin{aligned} k_x &= \left(i_x + \sigma_x(i,j)\lfloor\frac{|j_x - i_x|_{\mathcal{T}}}{2}\rfloor\right) \pmod{\sqrt{n}} \\ k_y &= \left(i_y + \sigma_y(i,j)\lfloor\frac{|j_y - i_y|_{\mathcal{T}}}{2}\rfloor\right) \pmod{\sqrt{n}}. \end{aligned} \tag{5.14}$$

In the road map γ we have just constructed, the maximum flight distance is smaller than $\frac{\sqrt{n}}{2} + 1$ in L_1 distance. Therefore, according to Lemma 7, for any edge e in γ, $\mathsf{E}[W_e] \geqslant \eta/n^{1.5}$, where η is a non negative constant slightly smaller than $1 - \ln 2$. Thus, for each path γ_{ij} we have:

$$\begin{aligned} |\gamma_{ij}| &= \frac{1}{\pi(i)\mathsf{E}[W_{i,k}]} + \frac{1}{\pi(k)\mathsf{E}[W_{k,j}]} \\ &= n\left(\frac{1}{\mathsf{E}[W_{i,k}]} + \frac{1}{\mathsf{E}[W_{k,j}]}\right) \\ &\leq \frac{2n^2\sqrt{n}}{\eta}. \end{aligned} \tag{5.15}$$

We can now compute the Poincaré coefficient:

$$\kappa = \max_e \sum_{\gamma_{ij} \ni e} |\gamma_{ij}|\pi_i\pi_j = \frac{1}{n^2} \max_e \sum_{\gamma_{ij} \ni e} |\gamma_{ij}|. \tag{5.16}$$

To compute this sum, we need to count the number of paths γ_{ij} in the road map that use a given flight e. In our construction, we have balanced the traffic load over all the short flights so that a flight e belongs to at most 8 paths. Indeed, if a path contains flight e, then e is either the first or second flight. In the first case, by construction, the second flight has to be approximatively as long as e. Moreover, because of quantized grid effects, there are actually only 4 different possible flights a traveler in flight e might take as second flight (see Fig. 5.8). Repeating this argument in the case where e is the second flight, we then obtain that a flight e appears in at most 8 paths. Combining (5.15) and (5.16), we get:

$$\kappa \leq \frac{16}{\eta}\sqrt{n}.$$

As a result,

$$\lambda_2 \leq 1 - \frac{\eta}{16\sqrt{n}},$$

which yields Lemma 8. The proof is complete by using equation (5.3).

■

In the next Section, we generalize this proof from grids to regular geometric graphs. The approach will be the same but the detailed computations will be different. Also, the construction of the paths in the travel agency method will need some refinement.

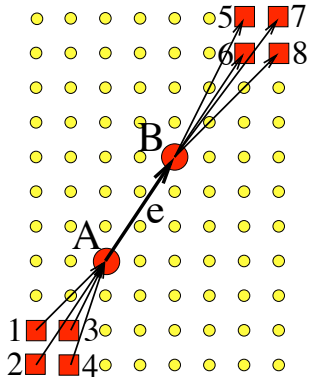


Figure 5.8: Number of paths including an edge $e = (A, B)$ in the road map. Paths have two hops of equal length, where equality here is defined up to grid effects. Therefore, for a given edge e, there are at most 8 paths including e: $(1, A, B)$, $(2, A, B)$, $(3, A, B)$, $(4, A, B)$ and $(A, B, 5)$, $(A, B, 6)$, $(A, B, 7)$, $(A, B, 8)$.

5.5.2 Performance of box-path averaging.

We now prove Theorem 6. All the fundamental ideas coming from the proof on grids in the previous section, appear here again, but sometimes in a more technical form. We have k boxes forming a torus grid as in the previous section and $k = \lceil\sqrt{(}n/(\alpha \log n))\rceil^2 \simeq n/(\alpha \log n)$, for some $\alpha > 2$.

Using regularity, each box contains a number of nodes between $a \log n$ and $b \log n$. We use the $(\leftrightarrow, \updownarrow)$-box routing scheme presented in Section 5.3.4. There are only a few modifications to make to the grid proof in order to obtain the regular geometric graph proof. The idea is to notice that for any route $r = (r_1, r_2, \cdots, r_\ell)$, we can attribute a box route $\widetilde{r}$ consisting of the boxes the nodes of r belong to. If we call $b(i)$ the box node i belongs to, then $\widetilde{r} = (b(r_1), b(r_2), \cdots, b(r_\ell))$. We call n_i the number of nodes in the box $b(i)$ node i belongs to. The sequence of n_i is fixed by the graph we are considering. ℓ_{ij} is the L_1 distance between boxes $b(i)$ and $b(j)$: $\ell_{ij} = \|b(j) - b(i)\|_1$. We denote by $\ell(r)$ the number of nodes in route r, $s(\widetilde{r})$ the starting box of route $\widetilde{r}$ and $d(\widetilde{r})$ its destination box. In our problem the chosen route is random, which we will denote by capital case letter: R, leading to other random variables $\widetilde{R}$, $\ell(R)$, $s(\widetilde{R})$, etc.

5.5.2.1 Evaluating $\mathsf{E}[W]$

Lemma 9. (Expected $\mathsf{E}[W]$ on the regular geometric graph) *For any pair of nodes (i, j) that do not belong to the same box, if their grid-distance normalized to the maximum grid-distance $\delta_{ij} = \ell_{ij}/\sqrt{k}$ is smaller than a constant, then*

$$\mathsf{E}[W_{ij}] = \Omega\left(\frac{1}{n\sqrt{n \log n}}\right). \tag{5.17}$$

More precisely,

$$\mathsf{E}[W_{i,j}] \geq \frac{4a}{b^2}\frac{2}{n^2}\sqrt{\frac{n}{\alpha \log n}}\left(1 - \delta_{ij} + \delta_{ij} \log \delta_{ij}\right), \tag{5.18}$$

Proof. For any node i and node j that do not belong to the same box, we want to compute the expectation of W_{ij}. Counting the routes in this setting is complicated because each sender has at least $a \log n$ nodes to send its message to. In order to use our simple analysis of the grid, we condition the expectation on the box routes $\widetilde{R}$. Given a box route, $W_{ij} = 0$ if i or j is not in the box route. On the contrary, if they both are in the box route, then $W_{ij} = 1/\ell(\widetilde{R})$ with probability $1/(n_i n_j)$. Indeed, if i (or j) is in starting box, the probability that i is the starting node is $1/n(i)$, because all the nodes wake up with the same rate. If i (or j) is in another box of the given

box route, then the probability that i is chosen is $1/n(i)$ as well, because the routing chooses next node uniformly among the nodes of the next box.

$$\begin{aligned}\mathsf{E}[W_{ij}] &= \mathsf{E}_{\widetilde{R}}[\mathsf{E}_R[W_{ij}|\widetilde{R}]] \\ &= \mathsf{E}_{\widetilde{R}}[\frac{1}{n_i n_j}\frac{1}{\ell(\widetilde{R})}1_{b(i)\in\widetilde{R}}1_{b(j)\in\widetilde{R}}].\end{aligned}$$

From now on, we are back to a problem with routes on a grid which has k "nodes". The difference with previous section is that routes are no longer uniform. Indeed, now, boxes wake up more frequently if they contain more nodes: the probability that box b_i wakes up is n_i/n. Destination boxes are still chosen uniformly at random with probability $1/k$ because there are k boxes in total. Just as before, we consider only $(\leftrightarrow, \updownarrow)$-box routes so that a box route is entirely determined by its starting box and its destination box, and we count box routes of different length separately. Let $\mathcal{R}_{ij}^{\ell}$ be the set of box routes of size ℓ including b_i and b_j.

$$\begin{aligned}\mathsf{E}[W_{ij}] &= \frac{1}{n_i n_j}\sum_{\widetilde{r}}\frac{1_{b(i)\in\widetilde{r}}1_{b(j)\in\widetilde{r}}}{\ell(\widetilde{r})}\mathbb{P}[\widetilde{R}=\widetilde{r}] \\ &= \frac{1}{n_i n_j}\sum_{\ell=\ell_{ij}+1}^{2\lfloor\frac{\sqrt{k}}{2}\rfloor+1}\sum_{\widetilde{r}\in\mathcal{R}_{ij}^{\ell}}\frac{\mathbb{P}[\widetilde{R}=\widetilde{r}]}{\ell} \\ &= \frac{1}{n_i n_j}\sum_{\ell=\ell_{ij}+1}^{2\lfloor\frac{\sqrt{k}}{2}\rfloor+1}\sum_{\widetilde{r}\in\mathcal{R}_{ij}^{\ell}}\frac{\mathbb{P}[s(\widetilde{R})=s(\widetilde{r}),d(\widetilde{R})=d(\widetilde{r})]}{\ell} \\ &= \frac{1}{n_i n_j}\sum_{\ell=\ell_{ij}+1}^{2\lfloor\frac{\sqrt{k}}{2}\rfloor+1}\sum_{\widetilde{r}\in\mathcal{R}_{ij}^{\ell}}\frac{1}{\ell}\frac{n_{s(\widetilde{r})}}{n}\frac{1}{k}.\end{aligned}$$

We now use the regularity of the graph : for any node m, $a \log n \leqslant n_m \leqslant b \log n$.

$$\begin{aligned}
\mathsf{E}[W_{ij}] &\geqslant \frac{1}{(b \log n)^2} \sum_{\ell=\ell_{ij}+1}^{2\lfloor \frac{\sqrt{k}}{2} \rfloor + 1} \frac{1}{\ell} \frac{a \log n}{n} \frac{4 \log n}{n} |\mathcal{R}_{ij}^{\ell}|. \\
&= \frac{4a}{b^2} \frac{1}{n^2} \sum_{\ell=\ell_{ij}+1}^{2\lfloor \frac{\sqrt{k}}{2} \rfloor + 1} \frac{|\mathcal{R}_{ij}^{\ell}|}{\ell} \\
&\geqslant \frac{4a}{b^2} \frac{2}{n^2} \left(\sqrt{k} - \ell_{ij} - \ell_{ij} \ln \frac{\sqrt{k}}{\ell_{ij}} \right).
\end{aligned}$$

The last inequality comes from the same computation as for the grid, and it can be reformulated as in Lemma 9 when using the normalized distance coefficient $\delta_{ij} = \ell_{ij}/\sqrt{k}$. ■

5.5.2.2 Bounding $\lambda_2(\mathsf{E}[W])$

Lemma 10 (Relaxation time RGG)**.**

$$\frac{1}{1 - \lambda_2(\mathsf{E}[W])} = O(\sqrt{n \log n}). \tag{5.19}$$

Proof. As for the grid, we now apply the travel agency method. The situation is very similar to the grid case, except that boxes now contain $\Theta(\log n)$ nodes each.
Similarly to the grid case, we will be using 2-hop paths for every pair of nodes, by adding one intermediate stop half-way. More precisely, this intermediate stop is chosen in the box whose coordinates on the underlying lattice are given by equations 5.14, where i and j are the lattice coordinates of the source and destination boxes. Then, within each box, we need to carefully and fairly assign the intermediate nodes because a flight should not be used more than a constant number of times (it was 8 for the grid), otherwise it would create congestion. It is not hard to design such road maps because the number of nodes in each box varies at most by a constant multiplicative factor b/a.

To show this, assume that each box contains exactly $\log n$ nodes. Then, there are $(\log n)^2$ road maps to find between all the nodes in a pair of boxes (assume box 1 and 3, and let box 2 be the one half-way), but happily enough, there are $(\log n)^2$ flights between box 1 and box 2 and also between box 2 and 3. Therefore, as we can see on Fig. 5.9, the box path (box 1, box 2, box 3) can correspond to $(\log n)^2$ node road maps all using different flights (edges). This flight allocation technique can easily be extended to cases where the boxes do not have the same number of airports by using

some flights at most $\lceil b/a \rceil$ times each in the paths between two given boxes.

There is a second refinement to the grid proof: solving the problem for nodes that

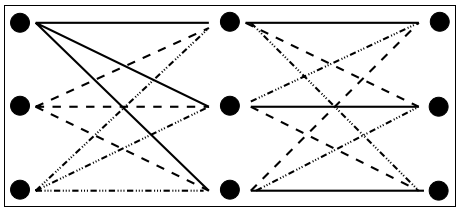


Figure 5.9: Path allocation when there are 3 nodes per box and thus 9 paths to design.

share a common box, which do not average jointly (Our bound on $\mathsf{E}[W_{ij}]$ is zero). However there are many edges to nodes in neighboring boxes. So formally, if node i and node j are in the same box, we design the road map from i to j to be a two hop road map stopping at a node located in the box above their box. By sharing fairly the available relay airports, the short north-south flights might be used in $\lceil b/a \rceil$ extra road maps.

We can thus construct road maps for any pair of airports that will use at most $9\lceil b/a \rceil$ times each good intensity flight. The rest of the proof is identical to the grid proof.

For each path we have:

$$\begin{aligned} |\gamma_{ij}| &= \frac{1}{\pi(i)\mathsf{E}[W_{i,k}]} + \frac{1}{\pi(k)\mathsf{E}[W_{k,j}]} \\ &= n\left(\frac{1}{\mathsf{E}[W_{i,k}]} + \frac{1}{\mathsf{E}[W_{k,j}]}\right) \\ &\leq cn^2\sqrt{n\log n}, \end{aligned} \tag{5.20}$$

for some constant c. Inequality 5.20 was obtained with the same reasoning as in the grid. We therefore conclude, using the Poincaré coefficient argument that

$$\kappa \leq 9\lceil \frac{b}{a} \rceil c\sqrt{n\log n}.$$

As a result, for n large enough, and some constant c'.

$$\lambda_2 \leq 1 - \frac{1}{c'\sqrt{n\log n}},$$

which yields the lemma. ■

5.5.3 Regularity of random geometric graphs

Lemma 11 (Regularity of random geometric graphs)**.** *Consider a random geometric graph with n nodes and partition the unit square in boxes of size $\alpha\frac{\log n}{n}$. Then, all the boxes contain $\Theta(\log n)$ nodes, with high probability as $n \to \infty$.*

Proof. Let X_i denote the number of nodes contained in the ith box. X_i are (non-independent) Binomially distributed random variables with expectation $\alpha \log n$. Standard Chernoff (we do not optimize for the constants) bounds [Motwani and Raghavan, 1995b] imply:

$$\mathbb{P}(X_i \leq \frac{\alpha}{2} \log n) \leq e^{-\alpha/8 \log n}.$$

and

$$\mathbb{P}(X_i \geq 2\alpha \log n) \leq e^{-\alpha/3 \log n}.$$

which give tight bounds on the number of nodes in each box:

$$\mathbb{P}(\frac{\alpha}{2} \log n \leq X_i \leq 2\alpha \log n) \geq 1 - 2e^{-\alpha/8 \log n}. \tag{5.21}$$

A union bound over boxes yields the uniform bounds on the maximum and minimum load of a square:

$$\mathbb{P}(\frac{\alpha}{2} \log n \leq \min_i X_i \leq \max_i X_i \leq 2\alpha \log n) \geq 1 - n^{1-\alpha/8} \frac{2}{\alpha \log n}.$$

Therefore, selecting $\alpha \geq 8$ yields the lemma. A more technical proof shows that the lemma holds for $\alpha > 2$. ■

Chapter 6

Conclusions and Future work

6.1 Distributed Storage

For storage problems, we think the most important conceptual contribution is the introduction of the information flow graph (Chapter 3). It is a graphical representation of how information is communicated across different storage nodes and time (through the links linking *in* and *out* copies of storage nodes). Information flow graphs can be created for any storage system even if it uses any hybrid of replication and coding since it abstracts the algebraic details of the encoded information and turns it into a flow problem. Cut-set arguments then yield information theoretic bounds on repair communication required and the network coding multicasting theorem shows that they can be achieved. The identified tradeoff between storage and bandwidth is essentially a tradeoff between communication over nodes (bandwidth) and communication across time (storage).

Certainly there are many issues that remain to be addressed before these ideas can be implemented in practical storage systems. In future work we plan to investigate deterministic designs of regenerating codes over small finite fields, the existence of *systematic* regenerating codes, designs that minimize the overhead storage of the coefficients, as well as the impact of node dynamics in reliability. Other issues of interest involve how CPU processing and disk I/O will influence the system performance, as well as integrity and security for the linear combination packets (see [C. Gkantsidis, 2006] for a related analysis for content distribution).

6.2 Distributed information processing

In Chapters 4 and 5 we introduced two novel gossip algorithms for distributed averaging. The proposed schemes operate in a distributed and asynchronous manne

on locally connected graphs and requires an order-optimal number of communicated messages for random geometric graph and grid topologies. The execution of path averaging requires that each node knows its own location, the locations of its nearest-hop neighbors and (for the routing-scheme that was theoretically analyzed) the total number of nodes n.

Location information is independently useful and likely to exist in many application scenarios. The key idea that makes path averaging efficient is the opportunistic combination of routing and averaging. The issues of delay (how several paths can be concurrently averaged in the network) and fault tolerance (robustness and recovery in failures) remain as interesting future work.

More generally, we believe that the idea of greedily routing towards a randomly pre-selected target (and processing information on the routed paths) is a very useful primitive for designing message-passing algorithms on networks that have some geometry. The reason is that the target introduces some directionality in the scheduling of message passing which avoids diffusive behavior. Other than computing linear functions, such path-processing algorithms can be designed for information dissemination or more general message passing computations such as marginal computations or MAP estimates for probabilistic graphical models. Scheduling the message-passing using some form of linear paths can accelerate the communication required for the convergence of such algorithms. Also understanding how node mobility or the physical wireless medium can be exploited to make more efficient distributed information processing algorithms are interesting open problems. We plan to investigate such research directions in future work.

Bibliography

[Acedanski *et al.*, 2005] S. Acedanski, S. Deb, M. Médard, and R. Koetter. How good is random linear coding based distributed networked storage. In *NetCod*, 2005.

[Ahlswede *et al.*, 2000a] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Trans. on Information Theory*, 46:1204–1216, 2000.

[Ahlswede *et al.*, 2000b] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Trans. Info. Theory*, 46(4):1204–1216, July 2000.

[Alanyali *et al.*, 2006] M. Alanyali, V. Saligrama, and O. Savas. A random-walk model for distributed computation in energy-limited networks. In *Proceedings of the 1st Workshop on Information Theory and its Applications*, San Diego, CA, 2006.

[Aldous and Fill, 2007] D. Aldous and J. Fill. Reversible markov chains on graphs. Book in preparation; available at http://www.stat.berkeley.edu/~aldous/RWG/book.html, 2007.

[Alon and Spencer, 2000] N. Alon and J. Spencer. *The Probabilistic Method*. Wiley Interscience, New York, 2000.

[Aysal *et al.*, 2008] T. C. Aysal, M. J. Coates, and M. G. Rabbat. Distributed average consensus with dithered quantization. *IEEE Transactions on Signal Processing*, To appear 2008.

[Bang-Jensen and Gutin, 2001] Jorgen Bang-Jensen and Gregory Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer, New York, 2001.

[Bash *et al.*, 2004] B.A. Bash, J.W. Byers, and J. Considine. Approximately uniform random sampling in sensor networks. In *Proc. of the 1st Workshop on Data Management in Sensor Networks (DMSN '04)*, August 2004.

[Benezit *et al.*, 2007] F. Benezit, A. G. Dimakis, P. Thiran, and M. Vetterli. Gossip along the way: Order-optimal consensus through randomized path averaging. In *Proceedings of Allerton Conference, Monticello, IL*, 2007.

[Bertsekas and Tsitsiklis, 1997] D. Bertsekas and J. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, Belmont, MA, 1997.

[Bhagwan *et al.*, 2004] Ranjita Bhagwan, Kiran Tati, Yu-Chung Cheng, Stefan Savage, and Geoffrey M. Voelker. Total recall: System support for automated availability management. In *NSDI*, 2004.

[Blake and Rodrigues, 2003] C. Blake and R. Rodrigues. High availability, scalable storage, dynamic peer neetworks: Pick two. In *Proc. HOTOS*, 2003.

[Blaum *et al.*, 1995] M. Blaum, J. Brady, J. Bruck, and J. Menon. EVENODD: An efficient scheme for tolerating double disk failures. *IEEE Trans. on Computing*, 44:192–202, February 1995.

[Bollobás, 2000] B. Bollobás. *Modern Graph Theory*. Springer-Verlag, New York, 2000.

[Bollobás, 2001] B. Bollobás. *Random Graphs (second edition)*. Cambridge University Press, 2001.

[Bolosky *et al.*, 2000] William J. Bolosky, John R. Douceur, David Ely, and Marvin Theimer. Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs. In *Proc. SIGMETRICS*, 2000.

[Boyd *et al.*, 2004] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Analysis and optimization of randomized gossip algorithms. In *Proceedings of the 43rd Conference on Decision and Control (CDC 2004)*, 2004.

[Boyd *et al.*, 2005] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms : Design, analysis and applications. In *Proceedings of the 24th Conference of the IEEE Communications Society (INFOCOM 2005)*, 2005.

[Boyd *et al.*, 2006] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, June 2006.

[Brémaud, 1999] P. Brémaud. *Markov Chains. Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer, 1999.

[C. Gkantsidis, 2006] P. Rodriguez C. Gkantsidis, J. Miller. Anatomy of a P2P content distribution system with network coding. *Proceedings of IPTPS*, 2006.

[Chen and G. Pandurangan, 2005] J.-Y. Chen and D. Xu G. Pandurangan. Robust aggregates computation in wireless sensor networks: Distributed randomized algorithms and analysis. In *2005 Fourth International Symposium on Information Processing in Sensor Networks*, 2005.

[Chun *et al.*, 2006] Byung-Gon Chun, Frank Dabek, Andreas Haeberlen, Emil Sit, Hakim Weatherspoon, M. Frans Kaashoek, John Kubiatowicz, and Robert Morris. Efficient replica maintenance for distributed storage systems. In *NSDI*, 2006.

[Dabek *et al.*, 2001] Frank Dabek, Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with CFS. In *Proc. ACM SOSP*, 2001.

[Dabek *et al.*, 2004] F. Dabek, J. Li, E. Sit, J. Robertson, M. Kaashoek, and R. Morris. Designing a dht for low latency and high throughput, 2004.

[Daskalakis *et al.*, 2007] C. Daskalakis, A.G. Dimakis, R. M. Karp, and M. J. Wainwright. Probabilistic analysis of linear programming decoding. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, January 2007.

[deGroot, 1974] M. H. deGroot. Reaching a consensus. *Journal of the American Statistical Association*, 69(345):118–121, March 1974.

[Denantes *et al.*, 2008] P. Denantes, F. Bénézit, P. Thiran, and M. Vetterli. Which distributed averaging algorithm should i choose for my sensor network? In *Proc. IEEE Infocom*, 2008.

[Denantes, 2007] Patrick Denantes. Performance of averaging algorithms in time-varying networks. Technical report, EPFL, 2007.

[Diaconis and Stroock, 1991a] P. Diaconis and D. Stroock. Geometric bounds for eigenvalues of Markov chains. *Ann. Applied Probability*, 1:36–61, 1991.

[Diaconis and Stroock, 1991b] P. Diaconis and D. Stroock. Geometric bounds for eigenvalues of markov chains. In *Annals of Applied Probability*, volume 1, 1991.

[Dimakis and Ramchandran, 2007] A.G. Dimakis and K. Ramchandran. Network coding for distributed storage in wireless networks. In *Networked Sensing Information and Control, Signals and Communication series*. Springer-Verlag, 2007.

[Dimakis *et al.*, 2005] A.G. Dimakis, V. Prabhakaran, and K. Ramchandran. Ubiquitous Acess to Distributed Data in Large-Scale Sensor Networks through Decentralized Erasure Codes. In *IEEE/ACM Int. Symposium on Information Processing in Sensor Networks (IPSN)*, April 2005.

[Dimakis *et al.*, 2006a] A.G. Dimakis, V. Prabhakaran, and K. Ramchandran. Decentralized erasure codes for distributed networked storage. In *IEEE Transactions on Information Theory*, June 2006.

[Dimakis *et al.*, 2006b] A.G. Dimakis, V. Prabhakaran, and K. Ramchandran. Distributed fountain codes for networked storage. In *Proceedings of IEEE ICASSP*, 2006.

[Dimakis *et al.*, 2006c] A.G. Dimakis, A. D. Sarwate, and M. Wainwright. Geographic gossip : Efficient aggregation for sensor networks. In *Proceedings of the Fifth International Symposium on Information Processing in Sensor Networks (IPSN 2006)*, Nashville, TN, April 2006.

[Dimakis *et al.*, 2006d] A.G. Dimakis, A.D. Sarwate, and M.J. Wainwright. Geographic gossip: Efficient aggregation for sensor networks. In *IEEE/ACM Int. Symposium on Information Processing in Sensor Networks (IPSN)*, 2006.

[Dimakis *et al.*, 2007] A.G. Dimakis, P.B. Godfrey, Y. Wu, M.J. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. In *Submitted for publication, preliminary version appeared in proceedings of IEEE Infocom*, 2007.

[Erdös and Sachs, 1963] P. Erdös and H. Sachs. Regulare graphen gegebene taillenweite mit minimaler knotenzahl. *Wiss. Z. Univ. Hall Martin Luther Univ. Halle–Wittenberg Math.– Natur.Reine*, 12:251–257, 1963.

[Fagnani and Zampieri, 2008] F. Fagnani and S. Zampieri. Randomized consensus algorithms over large scale networks. In *IEEE J. on Selected Areas of Communications, to appear*, 2008.

[Flajolet and Martin, 1985] P. Flajolet and G.N. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985.

[Fragouli *et al.*, 2006] C. Fragouli, J.Y. Le Boudec, and J. Widmer. Network coding: an instant primer. *ACM SIGCOMM Computer Comm. Review*, 2006.

[Gamal *et al.*, 2004] A. El Gamal, J. Mammen, B. Prabhakar, and D. Shah. Throughput-delay trade-off in wireless networks. In *Proceedings of the 24th Conference of the IEEE Communications Society (INFOCOM 2004)*, 2004.

[Ganesan *et al.*, 2004] D. Ganesan, R. Cristecu, and B. Beferull-Lozano. Power-efficient sensor placement and transmission structure for data gathering under distortion constraints. In *IEEE/ACM Int. Symposium on Information Processing in Sensor Networks (IPSN)*, April 2004.

[Godfrey *et al.*, 2006] P. Brighten Godfrey, Scott Shenker, and Ion Stoica. Minimizing churn in distributed systems. In *Proc. ACM SIGCOMM*, 2006.

[Guha *et al.*, 2006] Saikat Guha, Neil Daswani, and Ravi Jain. An experimental study of the Skype peer-to-peer VoIP system. In *IPTPS*, 2006.

[Gupta and Kumar, 2000] P. Gupta and P.R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, March 2000.

[Hafner, 2005] J. L. Hafner. WEAVER codes: Highly fault tolerant erasure codes for storage systems. In *FAST-2005: 4th Usenix Conference on File and Storage Technologies*, San Francisco, CA, December 2005.

[He *et al.*, 2003] T. He, C. Huang, B.M. Blum, J.A. Stankovic, and T. Abdelzaher. Range-free localization schemes for large scale sensor networks. In *Proceedings of the 9th Annual International Conference on Mobile computing and networking*, 2003.

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

[Ho *et al.*, 2006b] Tracey Ho, Muriel Médard, Ralf Koetter, David R. Karger, Michelle Effros, Jun Shi, and Ben Leong. A random linear network coding approach to multicast. *IEEE Trans. Inform. Theory*, 52(10):4413–4430, October 2006.

[Horn and Johnson, 1987] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, 1987.

[Huang and Xu, 2005] C. Huang and L. Xu. STAR: An efficient coding scheme for correcting triple storage node failures. In *FAST-2005: 4th Usenix Conference on File and Storage Technologies*, San Francisco, CA, December 2005.

[Huang *et al.*, 2007] C. Huang, M. Chen, and J. Li. Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems. In *IEEE International Symposium on Network Computing and Applications (NCA 2007)*, July 2007.

[Jaggi *et al.*, 2005] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen. Polynomial time algorithms for network code construction. *IEEE Trans. Inform. Theory*, 51:1973–1982, June 2005.

[Jiang, 2006] A. Jiang. Network coding for joint storage and transmission with minimum cost. In *International Symposium on Information Theory (ISIT)*, July 2006.

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

[Karp and Kung, 2000] B. Karp and H. Kung. Greedy perimeter stateless routing. In *Proceedings of ACM Conf. on Mobile Computing and Networking (MOBICOM), Boston, MA*, pages 243–254. ACM, 2000.

[Karp *et al.*, 2000] R. Karp, C. Schindelhauer, S. Shenker, and B. Vöcking. Randomized rumor spreading. In *Proc. IEEE Conference of Foundations of Computer Science, (FOCS)*, 2000.

[Katti *et al.*, 2006] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and Jon Crowcroft. XORs in the air: Practical wireless network coding. *Proc. of ACM SIGCOMM*, 2006.

[Kempe *et al.*, 2003] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proc. IEEE Conference of Foundations of Computer Science, (FOCS)*, 2003.

[Koetter and Médard, 2003] R. Koetter and M. Médard. An algebraic approach to network coding. *Transactions on Networking*, October 2003.

[Langendoen and Reijers, 2003] K. Langendoen and N. Reijers. Distributed localization in wireless sensor networks: a quantitative comparison. *Computer Networks*, 2003.

[Li and Dai, 2008] W. Li and H. Dai. Location-aided fast distributed consensus. In *IEEE Transactions on Information Theory, submitted*, 2008.

[Li *et al.*, 2003] S.-Y. R. Li, R. W. Yeung, and N. Cai. Linear network coding. *IEEE Trans. on Information Theory*, 49:371–381, February 2003.

[Lin *et al.*, 2007] Y. Lin, B. Liang, and B. Li. Data persistence in large-scale sensor networks with decentralized fountain codes. In *Proceedings of IEEE Infocom*, 2007.

[Luby *et al.*, 2001] M. Luby, M. Mitzenmacher, M.A. Shokrollahi, and D. Spielman. Improved low-density parity check codes using irregular graphs. *IEEE Trans. Info. Theory*, 47:585–598, February 2001.

[Luby, 2002] M. Luby. LT codes. *Proc. IEEE Foundations of Computer Science (FOCS)*, 2002.

[Lun *et al.*, 2006] D.S. Lun, N. Ratnakar, M. Médard, R. Koetter, D.R. Karger, T. Ho, E. Ahmed, and F. Zhao. Minimum-cost multicast over coded packet networks. *IEEE Transactions on Information Theory*, June 2006.

[Moallemi and van Roy, 2006] C. C. Moallemi and B. van Roy. Consensus propagation. *IEEE Trans. Info. Theory*, 52(11):1–13, 2006.

[Mosk-Aoyama and Shah, 2005] D. Mosk-Aoyama and D. Shah. Information dissemination via gossip: Applications to averaging and coding. http://arxiv.org/cs.NI/0504029, April 2005.

[Mosk-Aoyama and Shah, 2006] D. Mosk-Aoyama and D. Shah. Computing separable functions via gossip. In *Proceedings of the Twenty-Fifth ACM Symposium on Principles of Distributed Computing.*, Denver, CO, July 2006.

[Motwani and Raghavan, 1995a] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, Cambridge, UK, 1995.

[Motwani and Raghavan, 1995b] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, Cambridge, 1995.

[Nedic *et al.*, 2007] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. Tsitsiklis. On distributed averaging algorithms and quantization effects. In *LIDS Technical Report 2778, MIT,LIDS, submitted for publication*, 2007.

[Penrose, 2003] M. Penrose. *Random Geometric Graphs*. Oxford studies in probability. Oxford University Press, Oxford, 2003.

[Petrović *et al.*, 2006] D. Petrović, K. Ramchandran, and J. Rabaey. Overcoming untuned radios in wireless networks with network coding. *IEEE Transactions on Information Theory*, June 2006.

[Plank and Thomason, 2004] J.S. Plank and M.G. Thomason. A practical analysis of low-density parity-check erasure codes for wide-area storage applications. In *International Conference on Dependable Systems and Networks*, 2004.

[Pradhan and Ramchandran, 2003] S. S. Pradhan and K. Ramchandran. Distributed source coding using syndromes (DISCUS): Design and construction. *IEEE Trans. Info. Theory*, 49(3):626–643, 2003.

[Rabbat *et al.*, 2005] M. Rabbat, R. Nowak, and J. Bucklew. Robust decentralized source localization via averaging. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal processing (ICASSP)*, Philadelphia, PA, March 2005.

[Rabbat *et al.*, 2006a] M. Rabbat, J. Haupt, A.Singh, and R. Nowak. Decentralized compression and predistribution via randomized gossiping. In *ACM/IEEE Conference on Information Processing in Sensor Networks (IPSN'06)*, April 2006.

[Rabbat *et al.*, 2006b] M. Rabbat, J. Haupt, A. Singh, and R. Nowak. Decentralized compression and predistribution via randomized gossiping. In *IEEE/ACM Int. Symposium on Information Processing in Sensor Networks (IPSN)*, 2006.

[Ramamoorthy *et al.*, 2004] A. Ramamoorthy, K. Jain, P.A. Chou, and M. Effros. Separating distributed source coding from network coding. In *42nd Allerton Conference on Communication, Control and Computing*, 2004.

[Reed and Solomon, 1960] I.S. Reed and G. Solomon. Polynomial codes over certain finite fields. In *Journal of the SIAM*, 1960.

[Rhea *et al.*, 2001] S. Rhea, C. Wells, P. Eaton, D. Geels, B. Zhao, H. Weatherspoon, and J. Kubiatowicz. Maintenance-free global data storage. *IEEE Internet Computing*, pages 40–49, September 2001.

[Rhea *et al.*, 2003] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz. Pond: the OceanStore prototype. In *Proc. USENIX File and Storage Technologies (FAST)*, 2003.

[Rodrigues and Liskov, 2005] R. Rodrigues and B. Liskov. High availability in DHTs: Erasure coding vs. replication. In *Proc. IPTPS*, 2005.

[Rowstron and Druschel, 2001] A. Rowstron and P. Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In *Proc. ACM SOSP*, 2001.

[Saligrama *et al.*, 2006] V. Saligrama, M. Alanyali, and O. Savas. Distributed detection in sensor networks with packet losses and finite capacity links. *IEEE Transactions on Signal Processing*, to appear, 2006.

[Sander *et al.*, 2003] P. Sander, S. Egner, and L. Tolhuizen. Polynomial time algorithms for network information flow. In *Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 286–294, San Diego, CA, June 2003. ACM.

[Sanghavi, 2007] Sujay Sanghavi. Intermediate performance of rateless codes. *Information Theory and Applications (ITA)*, 2007.

[Sarkar *et al.*, 2007] Rik Sarkar, Xianjin Zhu, and Jie Gao. Hierarchical spatial gossip for multi-resolution representations in sensor networks. In *Proc. of the International Conference on Information Processing in Sensor Networks (IPSN'07)*, pages 420–429, April 2007.

[Saroiu *et al.*, 2002] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proc. MMCN*, San Jose, CA, USA, January 2002.

[Scaglione, 2007] A. Scaglione. On the wireless communication architecture for consensus problems. In *Information Theory and Applications (ITA)*, 2007.

[Shokrollahi, 2006] A. Shokrollahi. Raptor codes. *IEEE Trans. on Information Theory*, June 2006.

[Sinclair, 1992] A. Sinclair. Improved bounds for mixing rates of markov chains and multicommodity flow. In *Combinatorics, Probability and Computing*, volume 1, 1992.

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

[Spanos *et al.*, 2005] D. Spanos, R. Olfati-Saber, and R. Murray. Distributed Kalman filtering in sensor networks with quantifiable performance. In *2005 Fourth International Symposium on Information Processing in Sensor Networks*, 2005.

[Stribling,] Jeremy Stribling. Planetlab all pairs ping. http://infospect.planetlab.org/pings.

[Tati and Voelker, 2006] K. Tati and G. M. Voelker. On object maintenance in peer-to-peer systems. In *Proc. IPTPS*, 2006.

[Tsitsiklis, 1984] J. Tsitsiklis. *Problems in decentralized decision-making and computation.* PhD thesis, Department of EECS, MIT, 1984.

[Wang *et al.*, 2006] D. Wang, Q. Zhang, and J. Liu. Partial network coding: Theory and application for continuous sensor data collection. *Fourteenth IEEE International Workshop on Quality of Service (IWQoS)*, 2006.

[Wang *et al.*, 2007] W. Wang, M. Garofalakis, and K. Ramchandran. Distributed sparse random projections for refinable approximation. In *IEEE/ACM Int. Symposium on Information Processing in Sensor Networks (IPSN)*, 2007.

[Weatherspoon and Kubiatowicz, 2002a] Hakim Weatherspoon and John D. Kubiatowicz. Erasure coding vs. replication: a quantitiative comparison. In *Proc. IPTPS*, 2002.

[Weatherspoon and Kubiatowicz, 2002b] Hakim Weatherspoon and John D. Kubiatowicz. Erasure coding vs. replication: a quantitiative comparison. In *Proc. IPTPS*, 2002.

[Weatherspoon *et al.*, 2005] H. Weatherspoon, Byung-Gon Chun, Chiu Wah So, and John Kubiatowicz. Long-term data maintenace in wide-area storage systems: A quantitative approach. Technical report, UC Berkeley, UCB/CSD-05-1404, July 2005.

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

[Wu and Li, 2006] C. Wu and B. Li. Echelon: Peer-to-peer network diagnosis with network coding. *Fourteenth IEEE International Workshop on Quality of Service (IWQoS)*, 2006.

[Wu *et al.*, 2007] Y. Wu, A. G. Dimakis, and K. Ramchandran. Deterministic regenerating codes for distributed storage. In *Allerton Conference on Control, Computing, and Communication*, Urbana-Champaign, IL, September 2007.

[Wu, 2006] Y. Wu. On constructive multi-source network coding. In *International Symposium on Information Theory (ISIT)*, July 2006.

[Xiao *et al.*, 2005] L. Xiao, S. Boyd, and S. Lall. A scheme for asynchronous distributed sensor fusion based on average consensus. In *2005 Fourth International Symposium on Information Processing in Sensor Networks*, 2005.

[Xu and Bruck, 1999] L. Xu and J. Bruck. X-code: MDS array codes with optimal encoding. *IEEE Trans. on Information Theory*, 45:272–276, January 1999.