Community Computation

by

Fulu Li

MSc., Media Arts and Sciences,
Massachusetts Institute of Technology (2005)

Submitted to the Program of Media Arts and Sciences,
School of Architecture and Planning,
In partial fulfillment of the requirement for the degree of

DOCTOR OF PHILOSOPHY
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2009

Signature of Author [illegible]
Program in Media Arts and Sciences
May 1, 2009

Certified by [illegible]
Andrew B. Lippman
Senior Research Scientist of Media Arts and Sciences
Program in Media Arts and Sciences
Thesis Supervisor

Accepted by [illegible]
Deb K. Roy
Chairman, Committee on Graduate Studies
Program in Media Arts and Sciences

Community Computation

by

Fulu Li

Submitted to the Program of Media Arts and Sciences,
School of Architecture and Planning,
On May 1, 2009 in partial fulfillment of the
requirement for the degree of
Doctor of Philosophy

Abstract

In this thesis we lay the foundations for a distributed, community-based computing environment to tap the resources of a community to better perform some tasks, either computationally hard or economically prohibitive, or physically inconvenient, that one individual is unable to accomplish efficiently. We introduce community coding, where information systems meet social networks, to tackle some of the challenges in this new paradigm of community computation.

We design algorithms, protocols and build system prototypes to demonstrate the power of community computation to better deal with reliability, scalability and security issues, which are the main challenges in many emerging community-computing environments, in several application scenarios such as community storage, community sensing and community security. For example, we develop a community storage system that is based upon a distributed P2P (peer-to-peer) storage paradigm, where we take an array of small, periodically accessible, individual computers/peer nodes and create a secure, reliable and large distributed storage system. The goal is for each one of them to act as if they have immediate access to a pool of information that is larger than they could hold themselves, and into which they can contribute new stuff in a both open and secure manner. Such a contributory and self-scaling community storage system is particularly useful where reliable infrastructure is not readily available in that such a system facilitates easy ad-hoc construction and easy portability. In another application scenario, we develop a novel framework of community sensing with a group of image sensors. The goal is to present a set of novel tools in which software, rather than humans, examines the collection of images sensed by a group of image sensors to determine what is happening in the field of view. We also present several design principles in the aspects of community security. In one application example, we present community-based email spam detection approach to deal with email spams more efficiently.

Thesis Supervisor: Andrew B. Lippman
Title: Senior Research Scientist of Media Arts and Sciences

Community Computation

by

Fulu Li

Thesis Committee:

Advisor: [illegible]

Andrew B. Lippman
Senior Research Scientist of Media Arts and Sciences
Massachusetts Institute of Technology

Thesis Reader: [illegible]

Joseph A. Paradiso
Associate Professor of Media Arts and Sciences
Massachusetts Institute of Technology

Thesis Reader: [illegible]

V. Michael Bove Jr.
Principal Research Scientist of Media Laboratory
Massachusetts Institute of Technology

Thesis Reader: [illegible]

Ramesh Raskar
Associate Professor of Media Arts and Sciences
Massachusetts Institute of Technology

Acknowledgements

I would like to thank my advisor Dr Andrew B. Lippman for his mentorship and tremendous support. I greatly appreciate his openness, challenge and understanding, which made my research at MIT enjoyable. I would like to thank my thesis committee members: Dr Joseph Paradiso, Dr V. Michael Bove Jr., and Dr Ramesh Raskar, whose warm support and technical insights have greatly improved the quality of this dissertation. I would also like to thank Dr David P. Reed for his technical insights and encouragement. I would also like to thank Henry Holtzman for his insightful comments.

I would like to thank Deborah Widener, Sandy Sener, Linda Peterson, Aaron Solle, Paula Aguilera, and all other dedicated Media Lab staff, folks at Necsys and facilities at MIT Media Lab, folks at MIT ISO for their various assistance and support.

I would also like to thank my collaborators outside of my group: Jim Barabas, Mo-han Hsieh, Ramesh Raskar, Ana Santos, Ankit Mohan, Robert Gens, whose collaborative spirits have always made the research work full of fun.

My friends and colleagues at MIT made the last four years memorable: Jim Barabas, David Gauthier, Charles Amick, Brian Wong, David Merrill, Quinn Smithwick, Ahmed Kirmani, Ankit Mohan, Douglas Lanman, Dawei Shen, Inna Koyrakh, Grace Woo, Angela Chang, Kwan Hong Lee, Nadav Aharony, Jaewoo Chung, Ana Santos, Anmol Madan, Shaundra Daily, Matt Hirsch, Marie-Jose Montpetit, Joe Dvorak, Phil Fleming, and all other people I have come to know in the US.

Finally, I am deeply grateful to the steadfast and unwavering support from my parents, my brothers, my sister, my wife, Jing, my son, Kevin, throughout the years. This dissertation is for Jing, Kevin and Kelly.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the growing popularity of the Internet, the expansion of wireless local and personal area networks, the ubiquity of mobile devices, nowadays people are living in an increasingly networked world. In general, a community consists of three pillars, which are connectivity, communication and coordination. Underpinned by a wide range of technologies that facilitate better connectivity, better communication and better coordination among people, the notion of community has taken on new meanings. In a sense, a community can be regarded as a group of connected things, composed of hardware, software and/or people and formed by common interests through activities.

With the evolution of digital world, we found ourselves surrounded by three clouds [168]: the first cloud is the connectivity utility such as the Internet, the second cloud is the resource utility such as Web 2.0, iPhones, etc., the third cloud is the interaction utility driven by social networks. The big question we want to address in this thesis is how to harness and unleash the power of a community and how to maximize the sum of the utilities of all the community members, given their heterogeneous resource constraints. The answer that we propose is by means of efficient community computation. We will address several aspects of community computation, which are community storage, community sensing and community security.

The central thread of this thesis is the community coding optimization framework that we introduced, where information systems meet social networks. Community coding is an emerging field of social computation and coding theory, which provides a method of aggregating resources from a community to maximize the sum of their utilities by allowing the community members to coordinate their contributions of resources based on some predefine rules such as proportional fairness and minimum utility rules.

With great popularity of peer-to-peer (P2P) community-based applications comes great challenge. One ancient three monks' story is particularly intriguing and inspiring. One monk has to shoulder two buckets of water by himself. Two monks can share the load, which is kind of nice. But add a third, no one wants to fetch the water any more due to the fact that every one expects the other two monks will do the job. This story tells us that making a concerted effort is the key for community computation and we need innovations for community-computing environments, which is fundamental for our society. We address the incentive issue in a community-computing environment by adopting a proportional fairness rule in the framework of community coding, where the more resources one contributes, the more utilities one is entitled to have.

In our daily lives, communities like to share things. People share photos on Flikr [169], share videos on YouTube [170], and even share bandwidth via BitTorrent [171]. People also like to help each other, reward each other and look after each other's welfare. In this thesis, we present a set of tools to help to tap the resources of a community for the benefits of all the community members. In one application scenario, we develop a community storage system that is based upon a distributed P2P (peer-to-peer) storage paradigm, where we take an array of small, periodically accessible, individual computers/peer nodes and create a secure, reliable and large distributed storage system. The goal is for each node to act as if they have immediate access to a pool of information that is larger than they could hold themselves, and into which they can contribute new stuff in a both open and secure manner. When there is a collection of materials such as the whole collection of American literature that is used by more than one person, it is far more efficient to use people as the redundant elements than it is to store duplicate copies of the information in each peer node's machine. We assume that peer nodes may enter and leave the e-library network randomly, and we make different parts of the e-library available, depending on which peer nodes are operating. Such a community storage system is particularly useful where reliable infrastructure is not readily available in that such a system facilitates easy ad-hoc construction and easy portability. One important property of such a community-based system is that every community member contributes resources to the whole community as a whole including bandwidth, storage and computing power, etc. Thus, as more members join the community, the total capacity of the system also increases accordingly. Another important property of such a system is its robustness against node failures due to its distributed nature.

In another application example, we develop a collaborative offline Wikipedia system where the information of the whole Wikipedia is spread using our novel erasure codes among a group of community members, whose online access to Wikipedia may not be available due to some sort of failure such as a natural disaster. Each community member is not only able to access the whole contents of Wikipedia by providing a small portion of the total required storage burden, they can also contribute new contents and share annotations of the articles among the community.

One basic principle that we recommend for a community-computing environment is to treat the system, consisting of a variety number of devices that belong to community members, not just as a data system or communication system, but also as a *human* system. This is clearly one of the major differences between RAID [172] (redundant array of inexpensive disks) and the community storage that we proposed. RAID is used for computer storage systems that can divide and replicate data among multiple hard disks to provide redundancy and reliability. The nature of RAID is just a collection of hard disks, which is dramatically different from the dynamic nature of community storage systems that involves with people and the community. For RAID, the collection of hard disks is static, which are much easier to manage. On the other hand, for community storage system, people can join or leave the network on the fly, which makes it much more challenging to manage. For RAID, the communication cost among those collection of hard disks are fixed, but for community storage system, the communication cost among community member nodes can change over time dramatically, due to the fluctuation of the congestion level of the network traffic and the nomadic nature of people who carry and control those devices. Typically, an array of disks in RAID are attached to a central machine or located within the same room or building, which could lead to correlated failures of those disks when the subnet covers that machine or room go down or when some accident like fire occurs to that room or building. On the other hand, community member nodes in the community storage system are widely distributed, which could gracefully migrate correlated failures in some region and still maintain its availability and functionalities.

Another basic principle that we recommend is to let the community members coordinate to collectively achieve cooperative welfares, security. This is clearly one of the major differences between relying on the 3rd party services and relying on community members to collectively provide services for themselves. Nowadays there are so many online services that are provided by third parties such as dropbox services for storage, etc. According to some statistics, almost 90% of the PC disks in today's corporate America are free. The question here is: can we take advantage of those free disk spaces and do something good? Community storage is the answer that we propose. In community storage system, the community members have the preference to store their contents at their trusted partners such as friends, families, colleagues, etc. It is a natural feeling that people would trust more in their friends, families rather than a third party that they have never encountered before and they may never encounter in the future. For the sake of contents security, one thing to keep in mind is not to put all the eggs in one basket. Community storage is a good example to realize this basic principle. Each node only stores partial encoded data object. If you lose your device, other people can not see the information on that device and you can also retrieve your contents back from other community member nodes. For the attackers, they have to compromise enough nodes in order to reconstruct the original data object.

The major contribution of this thesis is the introduction of a new field of community coding and the presentation of a novel erasure code of community codes. Community coding provides an optimization framework to maximize the total utilities of all community members given their heterogeneous resources constraints based upon some predefined rules such as proportional fairness and minimum utility rules. An erasure code transforms a data object of n blocks into more than n blocks such that only a subset of the encoded blocks is required to reconstruct the original data object. Erasure codes are widely used for a variety of applications such as data storage and reliable data communication to enhance availability and reliability by adding redundancy via coding. For example, in the case of community storage, the benefits to use such an erasure code is that only a subset of the stored pieces at other community member nodes are required to reconstruct the original data object. This can gracefully meet the challenge that it is not guaranteed that every community member is always available. In the case of P2P streaming for application level multicast, the use of community coding systematically maximizes the total number of received streaming content stripes among all community members given their incoming and outgoing bandwidth constraints. The proportional fairness component is seamlessly integrated in the community coding framework for P2P streaming. In addition, the use of erasure code in P2P streaming provides another level of flexibility and adaptability besides multiple description code (MDC) [62] in that the users can randomly select a subset of the stripes to reassemble the original media stream based upon the distribution of the dynamic traffic congestion levels in the network, the availability of the media contents at other community member nodes, and the incoming and outgoing bandwidth constraints at each community member node.

The main objective of this thesis is to lay the foundations to harness and unleash the power of a community by the presentation of the theory of community coding and a set of tools (algorithms, protocols, and prototypes) in several application areas such as community storage, community sensing and community security. The set of novel algorithms and protocols that we presented is to deal with reliability, security and scalability issues in several community-computing scenarios to tap the resources of a community to better perform some tasks, either computationally hard or economically prohibitive, or physically inconvenient, that one individual is unable to accomplish efficiently. We show its utility through actual prototypes.

Information storage and storage sharing are key components of modern networked computer systems. Most of the existing networked file systems or distributed storage systems are accomplished by the distribution of multiple file/storage servers. The distinguishing features of our proposed community storage system include the novel elements of community coding, the novel erasure code of community codes, and the novel aspects of P2P media sharing.

In another application scenario, we develop a novel framework of community sensing with a group of image sensors. The goal is to present a set of novel tools in which software, rather than humans, examines the collection of images sensed by a group of sensors to determine what is happening in the field of view.

We also present several design principles in the aspects of community security. In one application example, we present community-based email spam detection approach to deal with email spams much more efficiently.

The rest of the thesis is organized as follows. We present the theory of community coding in Chapter 2. The community storage prototypes are presented in Chapter 3. We present a set of analytical results, novel algorithms and protocols for community sensing in Chapter 4. We address different aspects of community security in Chapter 5. The contributions of this thesis are summarized in Chapter 6. We conclude this dissertation in Chapter 7.

Chapter 2

Community Coding

2.1 The Overview

To tackle some of the challenges in community computation, we introduce community coding, where information systems meet social networks. Community coding is an emerging field of social computation and coding theory, which provides a method of aggregating resources from a community to maximize the sum of their utilities by allowing the community members to coordinate their contributions of resources.

2.1.1 Community Coding

Formally, we introduce the notion of community coding as follows.

Let R_i stand for the available resources at node i in a community of N members. We define C_i as the committed resources from node i for a given joint event of the community. Let U_i denote the utilities for node i, resulted from the given event. Let U_F stand for the utility floor limit for members of the community.

The goal is to maximize the sum of the utilities among all the community members, subject to some given constraints.

$$\max_{C_i} \quad \sum_{i=1}^{N} U_i \tag{2.1}$$

Subject to

$$C_i \leq R_i \qquad \text{for } 1 \leq i \leq N \tag{2.2}$$

$$U_i \geq U_F \qquad \text{for } 1 \leq i \leq N \tag{2.3}$$

$$\frac{U_i}{U_j} \propto \frac{C_i}{C_j} \qquad \text{for } 1 \leq i, j \leq N \tag{2.4}$$

$$U_i = f(C_1, C_2, \ldots, C_i, \ldots, C_N) \qquad \text{for } 1 \leq i \leq N \tag{2.5}$$

Notably, the utility for each community member is not only determined by its committed resources but also by those contributed from other community members. Constraint 2.3 indicates the floor limit of the community member's utility, similar like the concept of minimum wage in our society. Constraint 2.4 denotes the proportional fairness issue in that the more resources one contributes, the more utilities that community member will be awarded.

2.1.2 Community Coding versus Network Coding

Network coding, which refers to coding at the nodes in a network [160], is a field of information theory and coding theory, which provides a method of attaining maximum information flow in a network.

The core notion of network coding is to allow mixing of data from multiple sources at intermediate network nodes to increase the entropy of the transmitted information. A receiver sees these data packets and deduces from them the messages that were originally intended for that data sink. Routing, which only routes information without mixing of the data at intermediate network nodes, can be regarded as a special case of network coding.

On the other hand, community coding is a field of social computation and coding theory, which provides a method of aggregating resources from a community to maximize the sum of their utilities by allowing the community members to coordinate their contributions of resources based upon some predefined rules such as the proportional fairness rule and the minimum utility rule.

In the following, we present several application scenarios of community coding, which are network-aware source coding with multiple sources, P2P streaming for application level multicast, and a novel erasure coding approach of community codes.

2.2. NASC with Multiple Sources

In this section, we present a network-aware source coding (NASC) approach for wireless broadcast channels with multiple sources in the framework of community coding. In this case, each source is regarded as a community member and the utility for each community member is the delivery of the information with the least total required resources. The proposed NSAC approach takes into account the information of the network topology during the source coding process and tends to maximize the sum of the utilities of all community members as a whole. We derive the optimal bounds for two scenarios: the minimization of the total traffic load and the minimization of the total required energy. Implementation issues are also discussed.

Traditional source coding schemes solely assume the role of data compression, e.g., the process of encoding information using fewer bits than an un-encoded representation. The proposed NSAC approach takes into account the information of the network topology and tends to maximize the sum of the utilities of all community members as a whole.

2.2.1 Related Work

Other researchers also address the source coding, power efficiency issues in wireless networks. The classic algorithm on source coding by Ziv and Lempel is described in [163]. The Huffman coding approach for deterministic sources is given in [76]. In [62], Goyal presented techniques for one single information source with several chunks of data, e.g., "descriptions", so that the source can be approximated from any subset of the chunks. The broadcast channel with degraded message set problem was described in multi-user information theory [37]. Bergmans present a similar broadcast channel scenario in [17]. Energy efficient issues for wireless ad hoc networks are described in [61] and [108]. Barr et al present an energy-aware lossless data compression approach in [15]. Scaglione and Serveto analyze the interdependence of routing and data compressing in wireless sensor networks in [129]. Zhao in [162] presents network source codes to take advantage of network topology for broadcast and multiple access systems. Li et al in [95] give an indepth analysis on the capacity of wireless ad hoc networks and observe that the throughput for each node's applications is limited by the relaying load imposed by distant nodes in the network.

2.2.2 Problem Formulation

We consider the situation that a sender wants to simultaneously transmit n independent messages from n distinct sources to n different destinations. Due to the popularity of multi-user and multi-task (process) operating system, the situation that a sender wants to send multiple messages simultaneously to different receivers could be commonplace in reality. Without loss of generality, we assume that the i^{th} source is destinated to the i^{th} receiver. We also assume that each message traverses different path with different path metrics, e.g., different number of hops, to its destination. The information from n different sources is multiplexed by the source encoder and the encoded information is further processed by the channel encoder and the modulator. The signal is propagated through radio channel. Each receiver receives its intended messages through the process of demodulation, channel decoding and source decoding. The communication structure is shown in Figure 2.1.

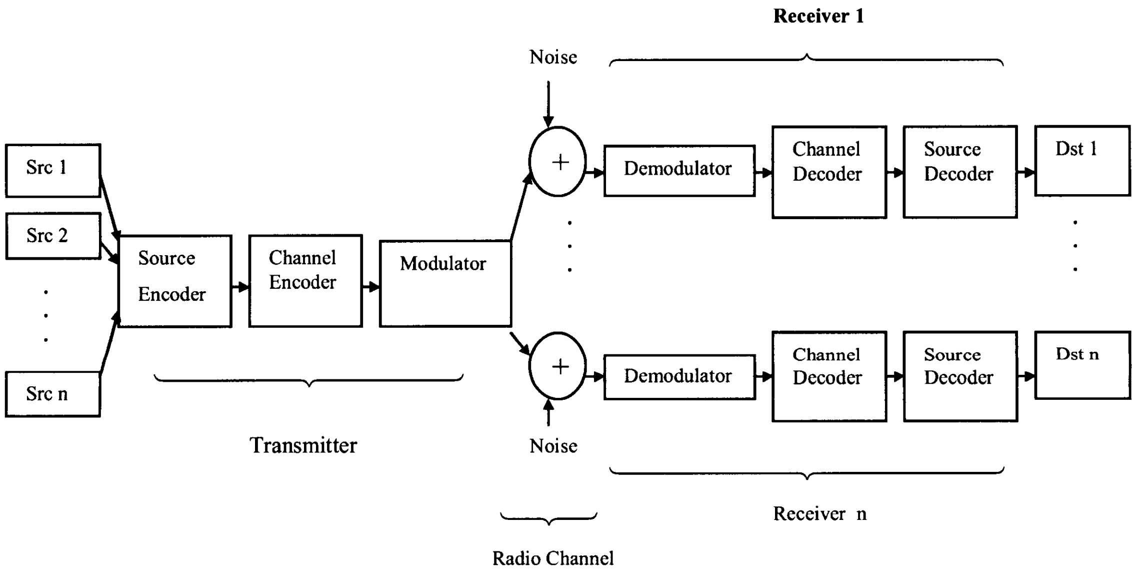


Figure 2.1: The communication system architecture.

We also assume the model of memoryless sources in the sense that the symbols for each message are generated independently of each other. Let s_i $(1 \le i \le n)$ stand for the number of independent symbols for the i^{th} source and $p_{i,j}$ $(1 \le i \le s_j,\ 1 \le j \le n)$ denote the probability of occurrence for the i^{th} symbol of the j^{th} source. Notably, for any source, say the i^{th} source, we have

$$\sum_{k=1}^{s_i} p_{k,i} = 1 \tag{2.6}$$

Without loss of generality, the minimization of the average codeword length is often used for the performance evaluation of a source encoder. From the network engineering perspective with a cross-layer design philosophy, we are most concerned with the total network throughput and the network lifetime. The total network throughput is determined by the total traffic load and the network lifetime largely depends on the energy consumption of each network node. Thus, the minimization of the average codeword length is not the only ultimate design object in our case. Rather, we consider the minimization of the total traffic load or the total required energy.

Let $L_i (1 \le i \le n)$ stand for the possible number of symbols in the i^{th} message and normally L_i is a large number. Then we can have the normalized probabilities among symbols in all possible sources. Let $p'_{i,j} (1 \le i \le s_j,\ 1 \le j \le n)$ denote the normalized probability of occurrence for the i^{th} symbol of the j^{th} source and we have

$$p'_{i,j} = \frac{p_{i,j} \times L_j}{\sum_{k=1}^{n} \sum_{m=1}^{s_k} (p_{m,k} \times L_k)} = \frac{p_{i,j} \times L_j}{\sum_{k=1}^{n} L_k} \tag{2.7}$$

$$\sum_{j=1}^{n}\sum_{i=1}^{s_j} p'_{i,j} = 1 \qquad (2.8)$$

Now let us recall the two basic principles on source coding, e.g., the prefix-free condition and the Kraft inequality. In order to guarantee instant decoding and to avoid confusion in the decoding process, the codeword of any symbol can not be the start part of the codeword of another symbol. The prefix-free condition guarantees unique decodability. It is known in this literature that any binary code satisfying the prefix-free condition, the codeword lengths $\{l_i\}$ must satisfy the Kraft inequality [55]. Let N stand for the total number of symbols and l_i denote the codeword length for the i^{th} symbol and the Kraft inequality constraint states that

$$\sum_{i=1}^{N} 2^{-l_i} \leq 1 \qquad (2.9)$$

Without considering the minimization of the traffic load or the total require energy, an extended Huffman coding approach based on the normalized probabilities among symbols in all the possible sources can be used to minimize the average codeword length. Let $l_{i,j}$ stand for the codeword length for the i^{th} symbol of the j^{th} source. Notably, the average codeword length in this case is downwards bounded by

$$\bar{l}_{\min} = -\sum_{j=1}^{n}\sum_{i=1}^{s_j} (p'_{i,j} \times l_{i,j}) \qquad (2.10)$$

where $l_{i,j} = -\log_2 p'_{i,j}$ (2.11)

2.2.3. NASC for Minimizing Total Traffic Load

We define the total traffic load to send the given n different messages from a given sender to n different destinations as the sum of the outgoing traffic, originated and forwarded/relayed traffic, at each node on the path from the sender to each destination node from the beginning of the transmission till all the n messages are completely delivered to the intended destination. Let $h_i\,(1 \le i \le n)$ denote the number of hops that the i^{th} source will traverse in order to reach its destination. Thus, we have the total traffic load

$$TL = \sum_{j=1}^{n} (L_j \times h_j \times \sum_{i=1}^{s_j} (l_{i,j} \times p'_{i,j})) \qquad (2.12)$$

Under the Kraft inequality constraint (Formula (2.9)), we apply Lagrange multiplier method to minimize the total traffic load (Eq. (2.12)) and we have

$$\begin{aligned} \min_{l_{i,j}} L = (&\sum_{j=1}^{n} (L_j \times h_j \times \sum_{i=1}^{s_j} (l_{i,j} \times p'_{i,j})) \\ &+ \lambda \times (\sum_{j=1}^{n} \sum_{i=1}^{s_j} 2^{-l_{i,j}} - 1) \end{aligned} \qquad (2.13)$$

Notably, in the above expression we consider L_j and $h_j\,(1 \le j \le n)$ as constants. By performing the minimization process using Lagrange multiplier method, we have

$$\frac{\partial L}{\partial l_{i,j}} = L_j \times h_j \times p'_{i,j} + \lambda \times 2^{-l_{i,j}} \times \ln 2 = 0$$

$$(1 \le j \ge n,\ 1 \le i \le s_j) \qquad (2.14)$$

$$\frac{\partial L}{\partial \lambda} = \sum_{j=1}^{n} \sum_{i=1}^{s_j} 2^{-l_{i,j}} - 1 = 0 \qquad (2.15)$$

From (2.6), (2.8), (2.14) and (2.15), we have

$$2^{-l_{i,j}} = -\frac{L_j \times h_j \times p'_{i,j}}{\lambda \times \ln 2} \tag{2.16}$$

$$\lambda = -\frac{\sum_{j=1}^{n}(L_j \times h_j \times \frac{L_j}{\sum_{k=1}^{n} L_k})}{\ln 2} = -\frac{\sum_{j=1}^{n}(L_j^2 \times h_j)}{\ln 2 \times \sum_{k=1}^{n} L_k} \tag{2.17}$$

From (2.16) and (2.17), we obtain that the traffic load gets its minimum when the following choice hold

$$l_{i,j} = -\log_2(L_j \times h_j \times p'_{i,j} \times \frac{\sum_{k=1}^{n} L_k}{\sum_{k=1}^{n}(L_k^2 \times h_k)}) \tag{2.18}$$

From Equation (2.18) we can see that the symbol with a higher occurring frequency should get a shorter codeword and the message that traverses more hops to its destination should get a shorter codeword. We can also see that the longer the message, e.g., more total number of symbols, the shorter the codewords for its symbols should be.

Let us define $\beta = \sum_{k=1}^{n} L_k \Big/ \sum_{k=1}^{n}(L_k^2 \times h_k)$.

Thus from (2.12) and (2.18), the total traffic load is downwards bounded by

$$TL_{\min} = \sum_{j=1}^{n}(L_j \times h_j \times \sum_{i=1}^{s_j}((-\log_2(L_j \times h_j \times p'_{i,j} \times \beta) \times p'_{i,j})) \tag{2.19}$$

In reality the total number of symbols in a message, say L_j for the j^{th} message may not be known *a priori*. We can relax this constraint by assuming the lengths of all the messages are approximately equal. With this relaxation, the minimization of the total traffic load is only bounded with the number of hops that a message traverses and the probability that a symbol occurs.

Equation (2.7) can be rewritten as

$$p'_{i,j} = \frac{p_{i,j}}{\sum_{j=1}^{n}\sum_{i=1}^{s_j} p_{i,j}} = \frac{p_{i,j}}{n} \tag{2.22}$$

Equation (2.18) can be rewritten as

$$l_{i,j} = -\log_2(p_{i,j} \times \frac{h_j}{\sum_{k=1}^{n} h_k}) \tag{2.21}$$

Equation (2.19) can be rewritten as

$$TL_{\min} = C \times \sum_{j=1}^{n}(h_j \times \sum_{i=1}^{s_j}((-\log_2(p_{i,j} \times \frac{h_j}{\sum_{k=1}^{n} h_k}) \times \frac{p_{i,j}}{n})) \tag{2.22}$$

where C is a constant, approximately representing the lengths of the messages.

2.2.4. NASC for Minimizing Total Required Energy

Although the path loss model, e.g., the power attenuation is proportional to $1/d^{\alpha}$ where d stands for the distance between the transmitter and receiver antennas and the exponent α often takes a value between 2 and 4, depending on the communication media characteristics, is often used to quantify the energy consumption in wireless communication protocols, it has been reported in [108] that the path loss model often fails to capture the energy overheads of the hardware. Following a similar model used in [72], we consider the radio dissipates E_{tr} nJ/bit to run transmitter or receiver circuitry and E_{amp} $pJ/bit/m^2$ for the transmit amplifier to achieve an acceptable E_b/N_o. We define the total required energy to send the given n different messages from a given sender to n different destinations as the sum of the required energy at each node on the path from the sender to each destination node from the beginning of the transmission till all of the n messages are completely delivered to the intended destination nodes. Let $l_{i,j}$ stand for the codeword length for the i^{th} symbol of the j^{th} source and $d_{i,j}$ denote the distance of the i^{th} hop on the path that the j^{th} message traverses. Thus, we have the total required energy

$$E = \sum_{j=1}^{n}((2\times h_j \times E_{tr} + E_{amp} \times \sum_{k=1}^{h_j} d_{k,j}^{\alpha}) \times L_j \times \sum_{i=1}^{s_j}(l_{i,j} \times p'_{i,j})) \qquad (2.23)$$

Under the Kraft inequality constraint (Formula (2.9)), we apply Lagrange multiplier method to minimize the total required energy (Eq. (2.23)) and we have

$$\begin{aligned} \min_{l_{i,j}} E = (&\sum_{j=1}^{n}((2\times h_j \times E_{tr} + E_{amp} \times \sum_{k=1}^{h_j} d_{k,j}^{\alpha}) \times L_j \times \sum_{i=1}^{s_j}(l_{i,j} \times p'_{i,j})) \\ &+ \lambda \times (\sum_{j=1}^{n}\sum_{i=1}^{s_j} 2^{-l_{i,j}} - 1) \end{aligned} \qquad (2.24)$$

Notably, in the above expression we consider L_j and $h_j (1 \le j \le n)$ as constants. By performing the minimization process using Lagrange multiplier method, we have

$$\frac{\partial E}{\partial l_{i,j}} = (2\times h_j \times E_{tr} + E_{amp}\times\sum_{k=1}^{h_j} d_{k,j}^{\alpha})\times L_j \times p'_{i,j}$$
$$+\lambda\times 2^{-l_{i,j}}\times \ln 2 = 0$$
$$(1\le j \ge n, 1\le i\le s_j) \tag{2.25}$$

$$\frac{\partial E}{\partial \lambda} = \sum_{j=1}^{n}\sum_{i=1}^{s_j} 2^{-l_{i,j}} - 1 = 0 \tag{2.26}$$

From (2.6), (2.8), (2.25) and (2.26), we have

$$2^{-l_{i,j}} = -\frac{(2\times h_j\times E_{tr} + E_{amp}\times\sum_{k=1}^{h_j} d_{k,j}^{\alpha})\times L_j\times p'_{i,j}}{\lambda\times\ln 2} \tag{2.27}$$

$$\lambda = -\frac{\sum_{j=1}^{n}((2\times h_j\times E_{tr} + E_{amp}\times\sum_{k=1}^{h_j} d_{k,j}^{\alpha})\times L_j \times \frac{L_j}{\sum_{k=1}^{n} L_k})}{\ln 2}$$
$$= -\frac{\sum_{j=1}^{n}((2\times h_j\times E_{tr} + E_{amp}\times\sum_{k=1}^{h_j} d_{k,j}^{\alpha})\times L_j^2)}{\ln 2\times\sum_{k=1}^{n} L_k} \tag{2.28}$$

From (2.27) and (2.28), we obtain that the total required energy gets its minimum when the following choice holds

$$l_{i,j} = -\log_2(L_j\times p'_{i,j}\times\frac{(2\times h_j\times E_{tr} + E_{amp}\times\sum_{k=1}^{h_j} d_{k,j}^{\alpha})\times\sum_{k=1}^{n} L_k}{\sum_{k=1}^{n}(L_k^2\times(2\times h_k\times E_{tr} + E_{amp}\times\sum_{m=1}^{h_k} d_{m,k}^{\alpha}))}) \tag{2.29}$$

From Equation (2.29) we can see that the symbol with a higher occurring frequency should get a shorter codeword and the message that traverses more hops over larger distances to its destination should get a shorter codeword. We can also see that the longer the message, e.g., more total number of symbols, the shorter the codewords for its symbols should be. Let us define

$$R_j = \frac{(2h_j E_{tr} + E_{amp}\sum_{k=1}^{h_j} d_{k,j}^{\alpha})\sum_{k=1}^{n} L_k}{\sum_{k=1}^{n}(L_k^2(2h_k E_{tr} + E_{amp}\sum_{k=1}^{h_k} d_{m,k}^{\alpha}))} \tag{2.30}$$

Thus from (2.23) and (2.29), the total required energy is downwards bounded by

$$E_{\min} = \sum_{j=1}^{n}((L_j(2h_j E_{tr} + E_{amp}\sum_{k=1}^{h_j} d_{k,j}^{\alpha})\sum_{i=1}^{s_j}(p'_{i,j}(-\log_2(L_j p'_{i,j} \times R_j))) \tag{2.31}$$

In reality the total number of symbols in a message, say L_j for the j^{th} message may not be known *a priori*. We can relax this constraint by assuming the lengths of all the messages are approximately equal. With this relaxation, the minimization of the total traffic load is only bounded with the number of hops that a message traverses and the probability that a symbol occurs.

Equation (2.7) can be rewritten as

$$p'_{i,j} = \frac{p_{i,j}}{\sum_{j=1}^{n}\sum_{i=1}^{s_j} p_{i,j}} = \frac{p_{i,j}}{n} \tag{2.32}$$

Equation (2.29) can be rewritten as

$$l_{i,j} = -\log_2(p_{i,j} \times \frac{2 \times h_j \times E_{tr} + E_{amp} \times \sum_{k=1}^{h_j} d_{k,j}^{\alpha}}{\sum_{k=1}^{n}(2 \times h_k \times E_{tr} + E_{amp} \times \sum_{m=1}^{h_k} d_{m,k}^{\alpha})}) \tag{2.33}$$

Equation (2.31) can be rewritten as

$$E_{\min} = C\sum_{j=1}^{n}((2h_j E_{tr} + E_{amp}\sum_{k=1}^{h_j} d_{k,j}^{\alpha})\sum_{i=1}^{s_j}((-\log_2(p_{i,j} \times R_j) \times \frac{p_{i,j}}{n})) \tag{2.34}$$

where C is a constant, approximately representing the lengths of the messages.

2.2.5. Summary

In this section, we present a network-aware source coding (NASC) approach for wireless broadcast channels with multiple sources in the framework of community coding. In this case, each source is regarded as a community member and the utility for each community member is the delivery of information with the minimal required resources. The proposed NSAC approach takes into account the information of the network topology and tends to maximize the sum of the utilities of all community members as a whole.

2.3. P2P Streaming for ALM

Recent advancement in peer-to-peer (P2P) technologies has enabled a wide range of new applications. In this section, we present community coding formulations to realize efficient P2P media streaming for application level multicast (ALM). In this case, each peer is regarded as a community member and the utility for each community member is the receiving of the streamed contents. Community coding is the key to simultaneously satisfy a large group (potentially hundreds) of peers' needs given limited network resources. The key to our approach is to cast the P2P media streaming problem as a community coding problem. In addition, we propose the use of erasure code in P2P streaming. The use of erasure code provides another level of flexibility and adaptability besides multiple description code (MDC) [62] in that the users can randomly select a subset of the stripes to reassemble the original media stream based upon the distribution of the dynamic traffic congestion levels in the network, the availability of the media contents at other community member nodes, and the incoming and outgoing bandwidth constraints at each community member node. We intend to answer the following question: given a source node, a group of intended destination peer nodes with heterogeneous network resources, what is the best way to distribute information among these peer nodes?

2.3.1. Introduction

In P2P streaming, the media streaming is accomplished in a peer-to-peer (P2P) fashion in the sense that each receiver node does not have to get the media content from the original media source, which may be far away geographically. Instead, each receiver node just needs to find if one of its peer neighbors is currently tuning in to receive the same media stream. Recursively, its neighbor will find its neighbor's neighbor all the way up to the original media source node.

Notably, P2P protocols/applications have received a great deal of attention recently in both academia and industry [16,35,41,77,101,102,120,161]. BitTorrent-like protocols [35] do not address the real-time needs of streaming as it is mainly designed to redistribute the file downloading and uploading cost among a group of peer nodes to achieve more robustness and scalability than traditionally centralized client-server model. P2P media streaming can be roughly classified into two categories: *(1)* P2P *on-demand* media streaming with *known* traffic profiles *a priori*; *(2)* P2P *live* media streaming with *uncertainty* of the *live* media stream traffic. The work in [41,120] assumes stored videos where the video traffic is known *a priori*. In live P2P media streaming, the dynamic traffic of the live media stream cannot be predicted accurately and one has to resort to *stochastic* models for robust network optimization [18].

The work in [41,120] on P2P video streaming do not consider application level multicast. In this section, we focus on application level multicast (ALM) [7,11,28,32,144,161] for P2P media streaming from one source to multiple intended peer destination nodes. Each peer node is equipped with network connectivity capability, which allows them to receive the data stream over the network and feeds the signal into the display devices.

In this section, we present community coding formulations in the framework of multiple interior-disjoint trees and mesh-based P2P streaming for application level multicast. The key to our analytical approach is to cast the P2P media streaming problem as a constraint system and to maximize the sum of the utilities of all community members. We intend to answer the following question: given a source node, a group of intended destination peer nodes with heterogeneous network resources, what is the best way to distribute information among these peer nodes?

2.3.2. Related Work

In the following, we give a general overview on application level multicast (ALM), SplitStream [28] and mesh-based P2P streaming, which are the basis for community coding formulations.

2.3.2.1 Application Level Multicast

While application level multicast (ALM) is a well-studied research area [7,11,28,32,144], it has to have some new ingredients in the realization of P2P streaming to satisfy the real-time needs in a distributed P2P fashion.

First, the ALM tree has to be constructed in such a way that media quality fluctuation is *minimized* among all peer viewers given limited network resources at each peer node. This is a particular challenge in the face of the uncertainty of the live media stream traffic.

Secondly, as discussed in [7], the ALM protocol has to operate well in *adversarial* scenarios. For example, the protocol must be able to deal with frequent node failures, rapid node joining and leaving (also called *churn*), denial of service (DoS) attacks, uncooperative peers, etc.

Thirdly, it is essential that the ALM system is contribution-aware as the contributions from the peer nodes are most likely to be *heterogeneous* due to the heterogeneous environments at each peer node in the real world. Contribution-awareness is also needed to establish meaningful charging models and incentive mechanisms in P2P media streaming.

2.3.2.2 SplitStream

The SplitStream scheme in [28] provides a multi-tree data delivery framework, which can leverage the distribution of the streaming data with *interior-node-disjoint* trees to better use the outgoing bandwidth among participating peer nodes. If $T1$ and $T2$ are two spanning trees which have no interior nodes in common then $T1$ and $T2$ are said to be *interior-node-disjoint*. Notably, each node in a set of trees is interior node in at most one tree and leaf node in the other trees [28].

Notably, high bandwidth rate for watching media stream may not be available consistently to some P2P clients from a set of peers. Using this SplitStream framework [28], peer nodes self-organize themselves into a forest of s trees, all rooted at media source node. The media source node encodes media content with source rate R evenly into s stripes of size R/S [144], each of which is distributed along a different tree. The low-rate stripes are re-combined upon playing-out at each destination node to obtain a high fidelity copy of the media content. The more stripes one receives, the better media quality it plays out. A layered codec based on multiple description coding (MDC) is typically used to realize this goal [25,144]. For example, Fine-Grained Scalable Coding (FGSC) is widely available in current video codec and is now part of the MPEG4 standard. FGSC is being increasingly used to encode videos in P2P networks [120].

2.3.2.3. Mesh-based P2P Streaming

As an alternative to the tree-based approaches, recent work in [101,102,161] present a mesh-based P2P streaming framework to better utilize outgoing bandwidth among participating peers. As the name suggests, participating peers initially form a directed mesh in mesh-based P2P streaming. Each peer node can have multiple parents and multiple child peers. Each peer node maintains a sufficient number of parents that can collectively fill its incoming link bandwidth, where each parent node provides a specific sub-stream of the content. In [101,102], the content delivery combines two phases: the diffusion phase along a diffusion tree (the black arrows), e.g., the push-based streaming from parents at lower level to child nodes at higher level (see Fig. 2.2), and the swarming phase (the red arrows), e.g., the pull-based streaming from parents at higher level to child nodes at lower level in a mesh-based overlay. As we can see from Figure 2.2 that the swarming phase maximizes the utilization of the outgoing bandwidth of the leaf nodes in the diffusion tree.

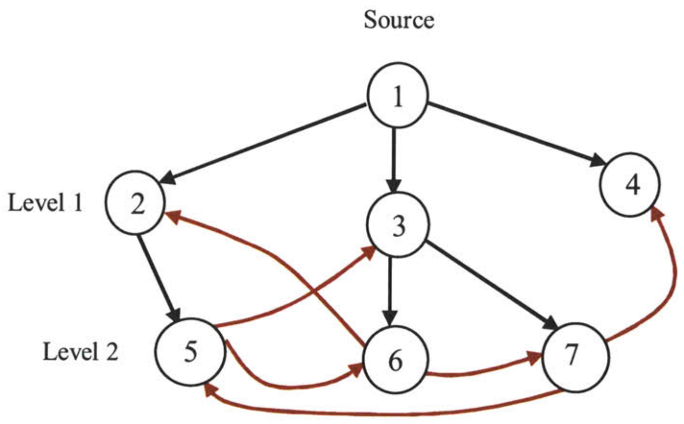


Figure 2.2: An illustration of mesh-based P2P streaming.

2.3.3. Community Coding Formulations

In this section, we present community coding formulations to realize P2P media streaming for application level multicast (ALM).

Let us define t_k as the k^{th} epoch in time ($0 \le k \le N$). Let $s_{i,k}$ be the number of stripes used at peer i in the playout of the k^{th} epoch t_k ($0 \le i \le n-1, 0 \le k \le N$). Let b_i^I be the incoming bandwidth limit at node $i\,(0 \le i \le n-1)$ and b_i^O be the outgoing bandwidth limit at node $i\,(0 \le i \le n-1)$. Let $f_{i,j,k}$ denote the number of stripes transmitted from node i to node j (no other peer nodes inbetween) during the k^{th} epoch t_k. Without loss of generality, we assume that the media source node is *Node* 0. We further define $x_{i,j,k}$ as the decision variable to indicate if there is a transmission from node i to node j (no other peer nodes inbetween) during the k^{th} epoch t_k. The variable $x_{i,j,k}$ is equal to one if there is a transmission from node i to node j (no other peer nodes inbetween) during the k^{th} epoch t_k, and zero otherwise. Let S denote the number of stripes when encoding the media source and R_k be the media source rate during the k^{th} epoch t_k. Let us define a decision variable x_e, which is equal to one if edge e is included in the tree, and zero otherwise. Let $abs(f(x))$ denote the absolute value of a function $f(x)$.

2.3.3.1 Multiple Interior-Disjoint Trees

In this section, we present a community coding formulation for the construction of multiple interior-disjoint multicast trees. The SplitStream scheme in [28] uses multiple *interior-node-disjoint* trees to leverage the distribution of P2P streaming to better use the outgoing bandwidth among participating peers.

NOTATION	
t_k	the k^{th} epoch in time
$s_{i,k}$	the number of stripes used at peer i in the playout of the k^{th} epoch t_k
b_i^I	the incoming bandwidth limit at node i
b_i^O	the outgoing bandwidth limit at node i
$f_{i,j,k}$	the number of stripes transmitted from node i to node j (no other peer nodes inbetween) during the k^{th} epoch t_k
$x_{i,j,k}$	the decision variable to indicate if there is a *direct* transmission from node i to node j during the k^{th} epoch t_k
S	the number of stripes when encoding the media source
x_e	the decision variable to indicate if edge e is included in the tree
R_k	the media source rate during the k^{th} epoch t_k

Table 2.1: Some notations for the community coding formulation of multiple interior-node-disjoint trees.

In Table 2.1, we summarize some of the notations used in the community coding formulation of the multiple interior-node-disjoint trees.

In the scenario of multiple interior-disjoint trees, $f_{i,j,k}$ takes a value of either one or zero as each stripe of the media source is distributed along a different interior-disjoint tree. We need to define two additional variables to distinguish constraints among different trees. We assume that variable $x_{i,j,k,s}$ is equal to one if there is a transmission from node i to node j (no other peer nodes inbetween) during the k^{th} epoch t_k along the s^{th} tree, and zero otherwise. Let us define a decision variable $x_{e,s}$, which is equal to one if edge e is included in the s^{th} ($1 \le s \le S$) tree, and zero otherwise.

Given a directed network graph $G = (V, E)$, where V is the vertex set and E is the edge set, the community coding problem can be formulated as a mixed integer programming (MIP) problem:

$$\text{maximize} \quad \sum_{i=1}^{n-1}\sum_{j=1}^{N} s_{i,j} \tag{2.35}$$

subject to:

$$\frac{R_k}{S} \times \sum_j f_{i,j,k} \le b_i^O \text{, for } 0 \le i \le n-1,\ 0 \le k \le N \tag{2.36}$$

$$\frac{R_k}{S} \times \sum_i f_{i,j,k} \le b_j^I \text{, for } 0 \le j \le n-1,\ 0 \le k \le N \tag{2.37}$$

$$s_{i,k} = \sum_j f_{j,i,k} \le S \text{, for } 0 \le i \le n-1,\ 0 \le k \le N \tag{2.38}$$

$$f_{i,j,k} \le 1 \text{, for } 0 \le i, j \le n-1,\ 0 \le k \le N \tag{2.39}$$

$$\sum_{i=0}^{n-1} x_{i,0,k,s} = 0 \text{, for } 0 \le k \le N, 1 \le s \le S \tag{2.40}$$

$$x_{i,i,k,s} = 0 \text{, for } 0 \le i \le n-1,\ 0 \le k \le N,\ 1 \le s \le S \tag{2.41}$$

$$\sum_{j=1}^{n-1} x_{0,j,k,s} \ge 1 \text{, for } 0 \le k \le N,\ 1 \le s \le S \tag{2.42}$$

$$f_{i,j,k} \le f_{m,i,k} \text{ if } x_{m,i,k,s} = 1 \text{, for } 1 \le j \ne i \le n-1,$$
$$0 \le k \le N, 1 \le s \le S \tag{2.43}$$

$$\sum_{i=0}^{n-1}\sum_{j=1}^{n-1} x_{i,j,k,s} = n-1, \text{ for } 0 \le k \le N, 1 \le s \le S \tag{2.44}$$

$$\sum_{s=1}^{S}(\min(\sum_{j=1}^{n-1} x_{i,j,k,s}, \quad 1)) = 1 \text{ , for } 0 \le i \le n-1, 0 \le k \le N \tag{2.45}$$

Now, let us define A_s as a non-empty set ($A_s \subset V$) and $E(A_s) = \{(i,j) \in E \mid (i,j) \in A_s\}$. We have the following constraint to eliminate all cycles:

$$\sum_{e \in E(A_s)} x_e \le |A_s| - 1, \text{ for } A_s \subset V, A_s \ne \Phi, V \tag{2.46}$$

$$\sum_{i=1}^{n-1}\sum_{j=1}^{N} abs(s_{i,j-1} - s_{i,j}) \le \eta \tag{2.47}$$

Our objective is to maximize the overall received stream data among all community members with constrained media playout quality fluctuation (Constraint 2.47) given limited network resources with multiple interior-disjoint trees. Constraint 2.36 indicates that the outgoing bandwidth has to be limited by the outgoing link capacity at each peer node. Likewise, Constraint 2.37 requires that the incoming bandwidth have to be limited by the incoming link capacity at each peer node. Constraint 2.38 and Constraint 2.39 indicate the number of stripes constraint for playout and traffic flow, respectively. Constraint 2.40 ensures that there is no flow to the root node (the media source node, *Node* 0) in each of the interior-disjoint trees. Constraint 2.41 indicates that no node transmits to itself. Constraint 2.42 means that at least there is one outgoing flow from the media source node (*Node* 0) in each of the interior-disjoint trees. Constraint 2.43 indicates that the downstream flow from a node is upper-bounded by incoming flow from its parent node. Constraint 2.44 means that there are exactly $n-1$ edges in each of the interior-disjoint trees (we have n nodes). Constraint 2.45 ensures that the final trees are interior-disjoint. Constraint 2.46 guarantees that there is no cycle in each of the final interior-disjoint trees. This is due to the fact that this constraint *recursively* guarantees that the number of included edges is less than or equal to the number of included vertices minus one, which eliminates any cycles in the tree [18]. Finally, Constraint 2.47 ensures that the media playout quality fluctuation among all peers is constrained and the parameter η can be set empirically based on user experiences.

2.3.3.2 Mesh-based P2P Streaming

As illustrated in Figure 2.2 that in mesh-based P2P streaming each participating peer node can have multiple parent nodes, which collectively fill its incoming link bandwidth, delivering as much sub-streams as possible to the given node. As described in [101,102], media data delivery is accomplished within two phases: the push-based diffusion phase and the pull-based swarming phase. We first consider the formation of the diffusion tree, rooted from the source node to reach every other participating peer node. Then we construct the swarming relationship among those peer nodes, where the swarming delivery operations only occur from a leaf node to the intermediate nodes or other leaf nodes in the diffusion tree.

Besides the notations defined earlier, we have the following additional definitions for mesh-based P2P streaming. Let $f_{i,j,k}^{T}$ denote the number of stripes transmitted from node i to node j (no other peer nodes inbetween) during the k^{th} epoch t_k along the diffusion tree. Likewise, let $f_{i,j,k}^{W}$ denote the number of stripes transmitted from node i to node j (no other peer nodes inbetween) during the k^{th} epoch t_k in the swarming delivery phase of operations. We assume that variable $x^{T}_{i,j,k}$ is equal to one if there is a transmission from node i to node j (no other peer nodes inbetween) during the k^{th} epoch t_k along the diffusion tree (see Fig. 2.2), and zero otherwise. Likewise, we assume that variable $x^{W}_{i,j,k}$ is equal to one if there is a transmission from node i to node j (no other peer nodes inbetween) during the k^{th} epoch t_k in the swarming delivery phase of operations (see Fig. 2.2), and zero otherwise. Let us define a decision variable $x_{e,T}$, which is equal to one if edge e is included in the diffusion tree, and zero otherwise.

To facilitate our discussion, we summarize some of the notations in the following table.

NOTATIOI	
$f^{T}_{i,j,k}$	the number of stripes directly transmitted from node i to node j during the k^{th} epoch t_k along the diffusion tree
$f^{W}_{i,j,k}$	the number of stripes directly transmitted from node i to node j during the k^{th} epoch t_k in the swarming delivery phase
$x^{T}_{i,j,k}$	the decision variable to indicate if there is a direction transmission from from node i to node j during the k^{th} epoch t_k along the diffusion tree
$x^{W}_{i,j,k}$	the decision variable to indicate if there is a direction transmission from from node i to node j during the k^{th} epoch t_k in the swarming delivery phase

Table 2.2: Some notations for the community coding formulation of mesh-based P2P streaming.

For the mesh-based P2P streaming, we have the community coding formulation as follows:

$$\text{maximize} \quad \sum_{i=1}^{n-1}\sum_{j=1}^{N} s_{i,j} \tag{2.48}$$

subject to:

$$\frac{R_k}{S}\times\sum_{j}(f^{T}_{i,j,k}+f^{W}_{i,j,k})\le b^{O}_{i}\text{ , for } 0\le i\le n-1,\ 0\le k\le N \tag{2.49}$$

$$\frac{R_k}{S}\times\sum_{i}(f^{T}_{i,j,k}+f^{W}_{i,j,k})\le b^{I}_{j}\text{ , for } 0\le j\le n-1,\ 0\le k\le N \tag{2.50}$$

$$s_{i,k}=\sum_{j}(f^{T}_{j,i,k}+f^{W}_{j,i,k})\le S\text{, for } 0\le i\le n-1,\ 0\le k\le N \tag{2.51}$$

$$f^{T}_{i,j,k}+f^{W}_{i,j,k}\le S\text{, for } 0\le i,j\le n-1,\ 0\le k\le N \tag{2.52}$$

$$\sum_{i=0}^{n-1}(x^{T}_{i,0,k}+x^{W}_{i,0,k})=0\text{, for } 0\le k\le N \tag{2.53}$$

$$x^{T}_{i,i,k}+x^{W}_{i,i,k}=0\text{, for } 0\le i\le n-1,\ 0\le k\le N \tag{2.54}$$

$$\sum_{j=1}^{n-1}x^{T}_{0,j,k}\ge 1\text{, for } 0\le k\le N \tag{2.55}$$

$$f^{T}_{i,j,k}\le f^{T}_{m,i,k}\text{ if } x^{T}_{m,i,k}=1\text{, for } 1\le j\ne i\le n-1,\ 0\le k\le N \tag{2.56}$$

$$\sum_{i=0}^{n-1}\sum_{j=1}^{n-1}x^{T}_{i,j,k}=n-1\text{, for } 0\le k\le N \tag{2.57}$$

Now, let us define A_T as a non-empty set ($A_T\subset V$) and $E(A_T)=\{(i,j)\in E\mid (i,j)\in A_T\}$. We have the following constraint to eliminate all cycles in the diffusion tree:

$$\sum_{e\in E(A_T)} x_e \le |A_T| - 1\text{, for } A_T \subset V, A_T \ne \Phi, V \tag{2.58}$$

$$\text{if } x^W_{i,j,k} = 1 \text{ , then } \sum_j x^T_{i,j,k} = 0,\ 1 \le j \ne i \le n-1,$$

$$0 \le k \le N \tag{2.59}$$

$$\text{if } x^W_{i,j,k} = 1 \text{ , then } x^T_{j,i,k} = 0,\ 1 \le j \ne i \le n-1,$$

$$0 \le k \le N \tag{2.60}$$

For the bandwidth per flow ratio constraint for each connection, we have:

if $x^W_{i,j,k} = 1$ or $x^T_{i,j,k} = 1$, then

$$abs\left(\frac{b^O_i}{\sum_l (x^T_{i,l,k} + x^W_{i,l,k})} - \frac{b^I_j}{\sum_l (x^T_{l,j,k} + x^W_{l,j,k})}\right) \le \gamma,$$

$$1 \le j \ne i \le n-1,\ 0 \le k \le N, \tag{2.61}$$

where γ is a given parameter.

$$\sum_{i=1}^{n-1}\sum_{j=1}^{N} abs(s_{i,j-1} - s_{i,j}) \le \eta \tag{2.62}$$

Our objective is to maximize the overall received stream data among all peers with constrained media playout quality fluctuation (Constraint 2.62) given limited network resources in a mesh-based P2P streaming framework. Constraint 2.49 indicates that the outgoing bandwidth has to be limited by the outgoing link capacity at each peer node. Likewise, Constraint 2.49 requires that the incoming bandwidth have to be limited by the incoming link capacity at each peer node. Constraint 2.51 and Constraint 2.52 indicate the number of stripes constraint for playout and traffic flow, respectively. Constraint 2.53 ensures that there is no flow to the root node (the media source node, *Node* 0) in both diffusion and swarming phases. Constraint 2.54 indicates that no node transmits to itself. Constraint 2.55 means that at least there is one outgoing flow from the media source node (*Node* 0) in the diffusion tree. Constraint 2.56 indicates that the downstream flow from a node is upper-bounded by incoming flow from its parent node in the diffusion tree. Constraint 2.57 means that there are exactly $n-1$ edges in the diffusion tree (we have n nodes). Constraint 2.58 guarantees that there is no cycle in the diffusion tree, which is part of the final mesh-based overlay. This is due to the fact that this constraint *recursively* guarantees that the number of included edges is less than or equal to the number of included vertices minus one, which eliminates any cycles in the tree. Constraint 2.59 indicates that the swarming delivery operation only occurs from a leaf node to other leaf nodes or intermediate nodes in the diffusion tree. Constraint 2.60 ensures that there is no swarming delivery operation from a child node to its parent node in the diffusion tree. Constraint 2.61 means that the bandwidth per flow ratio has to be roughly the same for each connection in the final mesh-based overlay. Finally, Constraint 2.62 ensures that the media playout quality fluctuation among all peers is constrained.

2.3.3.3 Contribution-Awareness

To provide tangible incentives to encourage peer nodes to increase their contributions, in [144] a contribution-aware overlay broadcast framework is presented to ensure that it distributes more bandwidth to nodes that contribute more.

$$\alpha \le \left. \sum_{k=0}^{t} s_{i,k} \middle/ \sum_{j=0}^{n-1}\sum_{k=0}^{t} f_{i,j,k} \right. \le \beta, \qquad 0 \le i \le n-1, 0 \le t \le N \qquad (2.63)$$

where α and β are threshold parameters to ensure contribution-awareness and the variable k denotes the k^{th} epoch in time.

2.3.4. Methods to Solve MIP

The most widely-used approaches to solve the integer programs (IPs) is to intelligently and efficiently search the solutions to the related linear programs (LPs) and check if the integer conditions are satisfied.

With appropriate transformations, the presented mixed integer programming formulations can be converted into the following standard matrix forms:

$$\begin{aligned} &\max \quad c'x + h'y \\ &\text{subject to:} \quad Ax + By = b \\ &\qquad x \in Z_{+}^{n}, \; y \geq 0 \end{aligned} \tag{2.64}$$

where x and y are the decision vectors, matrix A, B and vector b are determined by the constraints in the formulations.

After the transformation to the above standard matrix form, Gomory cutting plane algorithm or branch and bound method can be used to obtain the optimal solution based on the solutions for the corresponding linear programming (LP) problem.

An elastic constraint approach was presented in [111], aiming at increasing the possibility of finding the feasible solution of the problem. In [59], Glover outlined some key areas for integer programming including controlled randomization, learning strategies, induced decomposition and tabu search.

2.3.5. Summary

In this section, we present community coding formulations to realize efficient P2P media streaming for application level multicast (ALM). In this case, each peer is regarded as a community member and the utility for each community member is the receiving of the streamed contents. Community coding is the key to simultaneously satisfy a large group (potentially hundreds) of peers' needs given limited network resources. The key to our approach is to cast the P2P media streaming problem as a community coding problem. P2P media streaming may give rise to a wide range of new and exciting applications in the environment of community computation. We also discuss the means to obtain the optimal solution based on the community coding formulations. In addition, we propose the use of erasure code in P2P streaming. The use of erasure code provides another level of flexibility and adaptability besides multiple description code (MDC) [62] in that the users can randomly select a subset of the stripes to reassemble the original media stream based upon the distribution of the dynamic traffic congestion levels in the network, the availability of the media contents at other community member nodes, and the incoming and outgoing bandwidth constraints at each community member node.

2.4. Erasure Codes

Erasure code is special tool for community coding to provide redundancy and failure-resistance in various application scenarios among a community such as reliable data transmission and distributed data storage, etc.

2.4.1. Introduction

An erasure code is a mathematical transformation by which a data object of size S bytes is divided into n fragments of equal sizes (of S/n bytes), which are then encoded into $n+m$ total fragments and the reception of any n out of the $n+m$ total fragments suffice to recover the original data object [75,116,123]. The rate of the coding is determined by the quantity of $\frac{n}{n+m}$, which is less than one. Erasure codes can be roughly divided into two categories: fixed rate erasure code such as Reed-Solomon code [116,117,122], and rateless erasure code such as fountain code [26,44,45,98]. In general, an erasure code has a *rate* parameter, which indicates the transmitted information per bit for the encoded message. In other words, a rate denotes the fraction of the encoded output blocks required to reconstruct the original message [104]. An optimal erasure code with a rate of r transforms a message of n blocks into n/r blocks such that any n of those n/r blocks are sufficient to recover the original message. Rateless codes have the property that each message of size n has practically infinite encoding [104]. Essentially, the rate for rateless codes can asymptotically approach to zero. Notably, LT codes [98], online codes [104] and the presented community codes in this thesis are all rateless codes.

We say one code is locally encodable if any one encoding block can be computed quickly and independently of the other encoded blocks [104]. Similarly, we say one code is locally decodable if the given erasure codes have extremely efficient sublinear-time decoding algorithm [159].

2.4.2. Reed-Solomon Codes

Reed-Solomon codes [122] are well-known coding techniques for error correction in data transmission, fault tolerance in data storage systems, etc. For error correction, a Reed-Solomon code with m message symbols and $n = m + 2e$ code symbols can correct up to e errors [70]. The decoding algorithm for Reed-Solomon error-correction codes was discovered by Berlekamp and Welsh [70], who patented it in 1983.

Here we focus on the fault-tolerance (erasure) aspects of Reed-Solomon codes, where a data object is divided into n fragments of equal sizes, which are then encoded into $n+m$ total fragments and the availability of any n out of the $n+m$ total fragments suffice to recover the original data object. There are three main components of the Reed-Solomon algorithm for the implementation of erasure codes in RAID-like systems, which are the Vandermonde matrix to calculate and maintain checksum words, the Gaussian elimination to recover from failures, and the Galois Fields to perform arithmetic [116]. Notably, all the encoded blocks are calculated together, so Reed-Solomon codes are not locally encodable. All the decoded blocks are also calculated together and the computation complexity of the decoding process is not sublinear, so Reed-Solomon codes are not locally decodable.

2.4.3. LT Codes

LT codes (Luby transform codes) are the first class of practical fountain codes that are near optimal erasure codes invented by Michael Luby [98]. The distinctive feature of LT codes is its simple algorithm based on the exclusive OR operation (XOR) to encode and decode message. LT codes are rateless in that the encoding algorithm can in principle generate an infinite number of encode blocks. For the encoding process, it divides the message into n blocks, then it randomly pick $d\,(1 \le d \le n)$ of them to perform XOR operation to generate one encoded block. For the decoding process, it uses XOR operation recursively to retrieve the encoded message, starting from fully decoded blocks or the blocks with $d=1$. LT codes are locally encodable in that each encoded block is computed independently. LT codes are not locally decodable in that the computation complexity of the decoding process is not sublinear.

2.4.4. Raptor Codes

Raptor codes were invented by Amin Shokrollahi [134]. Raptor codes are one of the first known classes of fountain codes with linear time encoding and decoding. A distinctive feature of Raptor codes, compared with LT codes, is that Raptor codes are formed by the concatenation of two codes. The outer code is normally a fixed rate erasure code and the inner code is a form of LT code. The benefit of using LT code as an inner code is that the conditions for the decoding of the fixed rate erasure code, the outer code, can be relaxed. In general, concatenated codes are not a new idea and Thommesen addressed concatenated codes for error correction in [146] in 1987. For the decoding process, the inner code is decoded first.

2.4.5. Online Codes

In [104], Maymounkov and Mazieres present online codes, which are rateless codes and it is specifically designed for P2P multi-source big downloads. Online codes aim for the maximization of the utility of nodes with partial knowledge of a file to each other and the minimization of the bandwidth during the reconciliation phase to recover the original file. Online codes share a similar structure to that of LT codes [98] in that they both use the operation of XOR as the main computation operation for message blocks during the encoding and decoding process. LT codes have $O(\log n)$ encoding time and $O(n \log n)$ decoding time for a message of length n, while the encoding time and decoding time for online codes are $O(1)$ and $O(n)$, respectively. However, online codes require preprocessing and LT codes require no preprocessing. Online codes are rateless codes and they are also locally encodable.

2.4.6. Expander Codes

An expander graph is a graph in which every set of vertices has an unusually large number of neighbors [136]. One common way to determine if a particular graph is a good expander is to examine the gap between the largest and the second largest eigenvalues of the graph. If the second largest eigenvalue is far from the first, then the graph is a good expander [136].

Sipser and Spielman presented a family of error-correcting codes that are derived from expander graphs [136] and they termed those codes as expander codes. Expander codes belong to the class of low-density parity-check (LDPC) codes introduced by Gallager

[54] in early 1960s. To build an expander code, they begin with an unbalanced bipartite expander graph. Each of the n nodes on the large side of the graph is identified with one of the bits in a code of length n. They refer to these n bits as variables. Each of the vertices on the small side of the graph will be associated with a constraint on the variables. Each constraint indicates a set of linear restrictions on the variables that are its neighbors. The assumption is that those constraints are linearly independent. Despite the nice properties of expander codes in terms of encoding and decoding efficiencies, they are rarely used in practice due to the difficulty to verify the correctness of the constructed expander graph and the hypothesis of the linearly independent nature of the constraints.

2.4.7. Community Codes

We present a novel erasure code, termed as community codes. The distinguishing feature of community codes is its algorithm based on simple long binary arithmetic to encode and decode message. Another feature of community codes is that its encoding process involves mostly only two blocks and the expansion of distinct encoded block space is achieved by simple binary operations such as bit-shifting.

For the encoding process, it divides the message into n blocks, then it randomly pick $d (1 \leq d \leq 2)$ of them based on certain probability to perform simple long binary arithmetic to generate one encoded block. The probabilities to pick each block are reshuffled after the completion of each encoded block to ensure that distinct encoded blocks are generated. For the decoding process, it first looks for fully decoded blocks or the blocks with $d = 1$ or two encoded blocks that involve the same two original blocks.

Community codes are locally encodable in that each encoded block is computed independently. Community codes are locally decodable in that the computation complexity of the decoding process is sublinear due to the fact that on average the number of inquiries can be as small as two to retrieve one original block.

Chapter 3

Community Storage

3.1 The Overview

Information storage and storage sharing are key components of modern networked computer systems. In this chapter, we study collaborative storage in P2P systems, where contributory storage infrastructure is constructed among a group of peer nodes for a variety of emerging applications. We analyze the design tradeoffs of the collaborative storage system via concrete application examples in different settings. In this work, we focus on two main performance metrics: (1) the availability rate for a stored item in collaborative storage system when a peer node tries to access the given item, and (2) the uploading and retrieval delay for a stored data object of different sizes in a variety of circumstances. We experimentally verify the performance and feasibility of the constructed prototype of a shared e-library with a collection of e-books based upon the paradigm of collaborative storage in P2P systems, where a number of erasure-code encoded e-books are spread among a group of peer nodes.

With the ever-growing popularity of the Internet, the expansion of wireless local and personal area connectivity, the ubiquity of mobile devices, there is a tremendous social phenomenon of online sharing, either sharing video (like YouTube), pictures (like Flickr), webcam, or even bandwidth (like BitTorrent), for the benefits a large group of viewers/peers. Recent advancement in P2P (peer-to-peer) technologies has enabled a wide range of new applications based upon the online sharing of storage where contributory storage infrastructure is constructed among a group of peer nodes. In this chapter we study collaborative storage in P2P systems based on the needs of concrete application examples such as shared e-library with a collection of e-books, collaborative offline Wikipedia as well as file backup, etc.

Distributed P2P storage has attracted a great deal of research efforts recently [2,19,21,22,33,34,39,40,43-45,75,82,123,127,137]. In essence, distributed P2P storage is to build emerging networked storage services in a P2P fashion based upon contributory storage from a group of peer nodes in a networked computer system. One common challenge for such a system is to provide enough redundancy for the stored objects while minimizing the overall storage overhead. This is due to the fact that peer nodes may not be always online and storage devices like hard disks could fail. The most common practice to provide redundancy is accomplished by either replication or erasure code [2,33,44,45,75,82,117,123,125,127,137,154]. In a typical erasure code scheme, an original data object of size S bytes is split into n data fragments (often of the same size, e.g., S/n bytes) and a certain mathematical transform maps n data fragments into $n+m$ total fragments (n original data fragments and m redundant data fragments) such that any n encoded fragments out of the $n+m$ total fragments can recover the original data object [75,116]. We will elaborate more on erasure code in subsequent sections. There are a number of studies on the performance of replication vs. erasure code [19,125,154] and we will discuss more on this topic in Section 3.2. Nevertheless, replication can be viewed as a special form of erasure code of "1 out of R" if we have R replicas stored on peer nodes. One notable observation is that *unbalanced* storage may occur if some files are very large and some files are very small for the replication scheme if the operation unit is a file for storage on one peer node, compared with the erasure code approach.

In order for such a collaborative storage system to be highly useful, it has to guarantee a reasonable availability rate for a stored object when peer nodes are trying to access the given object. Here, the availability rate means the probability of the event that a peer node is able to successfully access a given data object stored in the distributed collaborative storage system. Moreover, the delay to upload an object to peer nodes or to retrieve a stored object from peer nodes has to be reasonably small, otherwise, the users may choose to renege from the system.

In this chapter, we examine the engineering design tradeoffs for such systems regarding availability rate and uploading/retrieval delay. We conduct quantitative and qualitative experiments to verify the performance and feasibility of the constructed prototype of a shared e-library with a collection of e-books based on the collaborative storage paradigm.

3.2. The Related Work

The early work by R. Anderson in [6] envisioned the construction of a storage medium for the eternity service with the property of being able to withstand denial of service attacks to safeguard the availability of the data. The basic idea is to use redundancy and scattering techniques to replicate data across a large set of machines and add anonymity mechanisms to drive up the cost of selective denial of service attacks to assure the availability of stored data.

In [43], Dimakis et al present methods on how to use random linear network coding to achieve the lower bounds on bandwidth that is required to maintain any distributed storage architecture. They also derive necessary and sufficient conditions on the amount of data that a newcomer has to download in order to generate a new erasure code fragment.

In [30], Chakrabarti et al discovered that the information survival threshold in a P2P network depends not only on the link and node fault probabilities but also on the largest eigenvalue of the connectivity matrix of the given P2P network. They use non-linear dynamical systems and fixed point stability theorems to derive a closed-form formula for the information survival threshold in P2P networks under dynamic network conditions with nodes' ups and downs. In [40], Datta and Aberer analyze the steady-state behavior of Internet-scale P2P storage system under churn (with rapid node joins and leaves) and they conclude that given a fixed rate of churn and a specific maintenance strategy, the system operates in a corresponding steady-state, e.g., dynamic equilibrium.

Availability issue for distributed storage systems was studied in [19,21,125]. In [19], Bhagwan et al presented *TotalRecall*, a P2P storage system that automates the task of availability management. The presented *TotalRecall* system automatically measures and estimates the availability of its constituent host components and predicts their future availability based on past behaviors. In [21], Blake and Rodrigues address reliable storage systems in P2P networks and the challenges they face. They conclude that large-scale cooperative storage is limited by likely dynamics and cross-system bandwidth, not by local disk space. In [125], Rodrigues and Liskov studied high availability issues in DHTs (distributed hash table) and compared two popular redundancy schemes: replication and erasure code. They conclude that in some cases the benefits of erasure coding are limited due to the complexity introduced by erasure coding to the system (the complexity of encoding/decoding process and the redundancy maintenance). In this work, we will particularly investigate the encoding/decoding delays introduced by Reed-Solomon code in a variety of circumstances for the applications that we are concerned with.

The availability of a stored item in the P2P storage paradigm is always *negatively* affected by disk failures. Recent study on disk failures [115] reveals that annual disk failure rate ranges from 1.7% to 8.6%, which is much higher than the listed annual disk failure rate of 0.88% by the manufactures. Another recent study on disk failures in [130] also found that annual disk replacement rate exceeds 1%, with 2-4% common and up to 13% on some systems. Both studies in [115,130] observe that time between failures exhibit strong auto-correlation and long-range dependence. Another work on latent sector errors in disk drives in [8] focused specifically on latent sector errors (errors that go undetected until the corresponding disk sectors are accessed) with large scale disk samples, , and analyze their implications on the design of reliability of storage systems. In our analysis of availability rate in P2P storage systems, we will consider the factor of disk failures.

The seminal work Pond in [123] was the first realization of the Oceanstore concepts presented in [82]. It touches many aspects of an Internet-scale, persistent data store including location-independent routing, Byzantine update commitment, push-based update of cached copies, etc. In [123], the authors also reported retrieval delay measurement but the erasure code encoding/decoding process is performed on a 42-machine cluster, which may not be a common practice for the applications that we are concerned with in this chapter. In this chapter, we report Reed-Solomon encoding and decoding delay measurements based on the process on individual peer nodes with different computing capabilities, e.g., different CPU speed and/or different memory space, for the shared e-library application that we are concerned with based upon the P2P collaborative storage paradigm.

The work on transparent file system (TFS) for contributory storage [34] aimed at removing barriers to contribution in P2P collaborative storage systems. TFS is designed to avoid impacting the performance of ordinary file access operations and provides significant improvement on storage efficiency compared with a user-space storage mechanism.

Rowstron and Druschel addressed storage management and caching in PAST [127], where a large-scale, P2P storage utility is constructed. PAST is built on top of P2P routing and lookup substrate and aims at scalability, self-organization, and persistence for Internet-scale P2P storage. In PAST system, redundancy is achieved by purely replicating data files, leaving the consideration of erasure code as its future directions. The cooperative file system (CFS) in [39] was intended as a file sharing medium. CFS is block-oriented and each block is stored on multiple nodes to provide redundancy. A conventional UNIX-like file system is layered on top of CFS. CFS is built on top of Chord (a P2P lookup mechanism based on DHT – distributed hash table), which maintains routing tables to find blocks. Another distributed P2P storage work based on DHT in [137] is intended for a cooperative off-site backup system. It uses Information Dispersal Algorithm (IDA) for its erasure code to provide redundancy. We refer interested readers to [127,39,137] for details due to limited space.

In [69], Haeberlen et al presented the storage system of Glacier to deal with correlated failures in distributed storage systems.

In this chapter, we focus on the availability rate analysis, uploading/retrieval delay analysis and experimental measurements for two emerging applications -- the shared e-library and collaborative offline Wikipedia in the framework of collaborative storage in P2P systems.

3.3. On the Choice of Erasure Codes

As discussed earlier, an erasure code is a mathematical transformation by which a data object of size S bytes is divided into n fragments of equal sizes (of S/n bytes), which are then encoded into $n+m$ total fragments [75,116,123]. The rate of the coding is determined by the quantity of $\frac{n}{n+m}$, which is less than one. Erasure codes can be roughly divided into two categories: fixed rate erasure code such as Reed-Solomon code [116,117,122], and rateless erasure code such as fountain code [26,44,45,98]. LT (Luby Transform) code is the first realization of fountain code. In the following discussion, we use LT code [98] as an example of rateless erasure code.

There are several subtle performance differences between fixed rate erasure code and rateless erasure code, which are: ***(a)*** for fixed rate erasure code like Reed-Solomon code, it is guaranteed that any n fragments out of the $n+m$ total encoded fragments suffice to recover the original data object; for rateless erasure code like LT code, it gives *probabilistic* guarantee that any n fragments out of the $n+m$ total fragments can recover the original data object in that on average a slightly more than n fragments out of the

$n+m$ total fragments are needed to recover the original data object; This is due to the fact that for LT code the $n+m$ total encoded fragments may not be distinct, depending on the degree distribution function [98]; ***(b)*** for fixed rate erasure code, all of the $n+m$ total encoded fragments are encoded together; while for rateless erasure code, each of the $n+m$ total encoded fragments is encoded independently; For rateless erasure code like LT code, it could generate infinite number of encoded fragments for a given data object because of the independent encoding nature for each encoded fragment, which may be an ideal property for packet/message recovery in communication system. ***(c)*** LT code (a rateless erasure code) employs a very simple algorithm based on XOR (exclusive OR) operation to encode and decode a data object; Reed-Solomon code (a fixed rate erasure code) uses Vandermonde matrix calculations, Galois field arithmetic as well as Gaussian Elimination techniques for encoding/decoding processes, which is much more complex than that of LT code. If the erasure code parameter $n+m$ and/or the data object size are very large, then the encoding/decoding process of Reed-Solomon code will be very slow. We will elaborate more on this in our experimental measurements for encoding/decoding delay of Reed-Solomon code with a number of e-books of different sizes in Section 6. ***(d)*** For LT code, a degree $d\,(1 \le d \le n)$ is chosen at random, which determines the number of fragments from the original n data fragments that are going to be XOR-ed to generate a new encoded fragment. The performance of LT code largely depends on the degree distribution function on which the random number of degrees is determined. Assume that a data object is divided into n fragments of equal sizes, which are then encoded into $n+m$ total fragments. The *maximum* number of *distinct* encoded fragments according to LT code based on the XOR operation among different original fragments is $2^n - 1$. If the condition of $(n+m \gg 2^n - 1)$ holds, then a lot of encoded fragments might happen to be *the duplication*, in which case the performance of LT code must deteriorate as it has to fetch much more than n encoded fragments to recover the original data object due to the duplication of encoded fragments. Notably, an erasure code requires n *distinct* fragments in order to recover the original data object.

As we discussed earlier in Chapter 2, the community codes that we presented are rateless, locally encodable and locally decodable.

3.4. Availability Analysis

We first give a brief overview on two of the emerging applications (shared e-library and collaborative offline Wikipedia) based on collaborative storage paradigm in P2P systems, which are the central threads of our discussions throughout this chapter.

3.4.1. Shared e-Library

Our vision for a shared e-library among a group of peer nodes is as follows. We take an array of small, periodically accessible, individual computers/peer nodes and create a secure, reliable and large distributed storage system. The goal is for each one of them to act as if they have immediate access to a pool of information, e.g., a shared e-library with a collection of e-books, that is larger than they could hold themselves, and into which they can contribute new stuff, e.g., uploading new e-books to the system, in a both open and secure manner. The more peer nodes you can reach, the larger the e-library you can access. When there is a collection of materials such as the whole collection of American literature that is used by more than one person, it is far more efficient to use people as the redundant elements than it is to store duplicate copies of the information in each peer node's machine. We assume that peer nodes may enter and leave the e-library network randomly, and we make different parts of the e-library available, depending on which peer nodes are operating.

There are different approaches to realize the above vision of a shared e-library. The first approach, say Approach A, is to store different e-books on different peer nodes. For this approach, anyone can read a given e-book provided that they can reach that peer node where that e-book is stored. The second approach, say Approach B, is to split any e-book into pieces and encode those pieces with erasure code to add redundancy and distribute those encoded pieces among a group of peer nodes. In this case (with Approach B), anyone can read a given e-book as long as a small group of peer nodes are reachable (the number of required peer nodes is determined by the erasure code parameters). In our prototype implementation, we adopted Approach B due to the following reasons: ***(a)*** for Approach A, if a peer requests an e-book that is stored at node X and node X is not currently reachable for this peer, then this peer is stuck; for approach B, even if node X is not reachable for this peer, he may still be able to fetch enough encoded segments for this e-book from other peer nodes and recover the given e-book; we call this "*higher availability*" advantage for Approach B. ***(b)*** For approach A, it will likely cause *unbalanced* storage if some e-books are very large and some e-books are very small. We call this "*better-balanced storage*" advantage for Approach B. ***(c)*** For approach A, if some e-book is very popular like "Harry Potter", then the peer node who stores the given e-book will be overwhelmed. For approach B, it leads to better network load balance as it spreads the load among several nodes that stored the encoded fragments of that e-book. We call this "*better network load balance*" advantage for Approach B.

According to M. Hart, the founder of project Gutenberg [75], an average US public library has about 30,000 books, which can be transformed into an e-library of 12 Gigabytes in zipped file format, assuming each e-book has one million characters. With the growing availability of e-books and the decreasing of the price of storage devices, it is possible for the people to own the civilization, instead of for the civilization to own the people [75]. With the advancement of collaborative storage techniques in P2P systems, it is very possible for the materialization of people's ambition to be the true possessor of the civilization even if an individual peer node may not have enough disk space.

3.4.2. Collaborative Offline Wikipedia

Wikipedia is an on-line encyclopedia with free content, written collaboratively by volunteers worldwide. It is probably one of the best examples of free information sharing for millions of people around the world so long as they have Internet access. However, the Internet connectivity may not be always available for rural countryside areas, places in poor countries, or for people who are on the road. Inspired by the offline Wikipedia project by T. Tsiodras [148], we consider the problem of collaborative offline Wikipedia, where the whole Wikipedia (of the size of 2.9 Gigabytes in a compressed form for the English version up to Aug. 27, 2007) are collaboratively stored over a group of peer nodes where the connectivity to the Internet is not available but the connectivity among the group of peer nodes is readily available, e.g., peer nodes equipped with WiFi capabilities can communicate among themselves in WiFi *ad hoc* mode. For example, if a group of friends go to mountain areas for an outing or a group of colleagues are on a business trip, it would be nice if they can still search information using collaborative offline Wikipedia platform with their laptops in WiFi *ad hoc* mode, provided that each of the laptops only needs to contribute a small amount of storage space (compared with more than 12 Gigabytes space requirement in uncompressed form for English version only if it is stored at one single machine).

Compared with the shared e-library application, collaborative offline Wikipedia presents unique challenges due to the much bigger size of the whole Wikipedia file compared with an e-book and the associated content search/inquiries within the entire Wikipedia.

3.4.3 Availability Analysis

In the following analysis, we assume that the total number of peers in the given P2P collaborative storage system is N. We also assume that a data object is divided into n fragments of equal sizes, which are then encoded into $n+m$ total fragments using Reed-Solomon code. As mentioned earlier, the availability rate means the *probability* of the event that a peer node is able to successfully access a given data object stored in the distributed collaborative storage system. Let p stand for the online probability of a peer node when other peers are trying to access data stored on this given peer node. Let f denote the disk failure probability. For the sake of simplicity, we also assume that so long as a peer node is online, the information stored on its dedicated space for collaborative storage is retrievable by other peers. We will not discuss the corresponding authentication and authorization mechanisms due to limited space.

We analyze availability rate in two different scenarios: Scenario A: not every peer has an encoded fragment for a given stored data object, i.e., $n+m<N$; Scenario B: every peer has an encoded fragment for a given stored data object, i.e., $n+m=N$.

3.4.3.1 Scenario A: $n+m<N$

For Scenario A in which *not* every peer has an encoded fragment for a given stored data object, i.e., $n+m<N$, the availability of a stored data item not only depends on which peers are available (only peer node with an encoded fragment of the given data item matters) but also on how many of these peers (with a distinct encoded fragment of the given data item) are available.

A. The shared e-library case:

Regarding the availability rate for a stored data item using Reed-Solomon erasure code with parameters of $(n, n+m)$ in the shared e-library case, we have the following closed-form formula:

$$Av\_rate = (1 - \sum_{i=0}^{m+n-1} (\binom{N}{i} \times (1-p)^{(N-i)} \times p^{i})) \times (1 - \sum_{j=0}^{n-1} (\binom{m+n}{j} ((1-f) \times p)^{j} \times (1-(1-f) \times p)^{m+n-j}))$$

where $2 \le n < N$, $1 \le m < N$, $m+n<N, 0 \le f, p \le 1$. (3.1)

The first part of the formula, e.g., $(1 - \sum_{i=0}^{m+n-1} (\binom{N}{i} \times (1-p)^{(N-i)} \times p^{i}))$, means that we need to find $n+m$ peers from the total number of N peers to store the corresponding encoded fragments for a given data object in the first place. The second part of the formula means that after the storage of the $n+m$ encoded fragments on $n+m$ peer nodes (one per peer node), we need to retrieve n fragments out of the total of $n+m$ encoded fragments in order to recover the original data object.

There are two possible reasons why a peer is not online: (a) its disk fails; (b) its disk is functional but it is not online. On the other hand, if a peer is online, its disk must be functional.

The numerical results for the shared e-library case are shown in Fig. 3.1~3.3, where n is equal to 10 and N is equal to 100.

When the condition of ($N >> n+m$) holds and for a reasonable value of p, say $p > 0.5$, the first part of the formula, e.g., (1 - $\sum_{i=0}^{m+n-1} (\binom{N}{i} \times (1-p)^{(N-i)} \times p^{i}$)), is very close to 1 and the 2nd part of the formula dominates the outcome.

We show the availability rate vs. different number of redundant fragments, e.g., m, in a variety of circumstances in Fig. 3.1~3.2. In Fig. 3.1, we assume that the peer online probability is 0.8. In Fig. 3.2, we assume that the peer online probability is 0.7. These values may be reasonable as most people have the similar kind of work schedule in the sense that when a peer is trying to access data stored on other peer nodes, they are mostly online due to the similar work schedule.

From Fig. 3.1~3.2, we can see that as m goes up, the availability rate for a given stored data item increases. This is due to the fact that a bigger m means more redundant fragments for a given stored data object. In particular, when the redundancy rate is up to around 80%, e.g., $m = 8$ and $n = 10$, the availability rate of a given stored data item is more than 99% for a reasonable disk failure rate, e.g., $f = 1\%$, and a reasonable peer online probability, e.g., p = 0.8. We also observe that the smaller the disk failure rate, the higher is the availability rate for a given stored data object. Compared with Fig. 3.1, the availability rate slightly goes down in Fig. 3.2 with the same settings due to a smaller peer online probability. High online probability of the peer nodes is the key for P2P collaborative storage applications when a peer tries to access data stored on other peer nodes. Otherwise, we have to increase the data redundancy. It is a tradeoff between storage space overhead and the online probability of peer nodes in order to achieve high data availability rate in P2P collaborative storage systems.

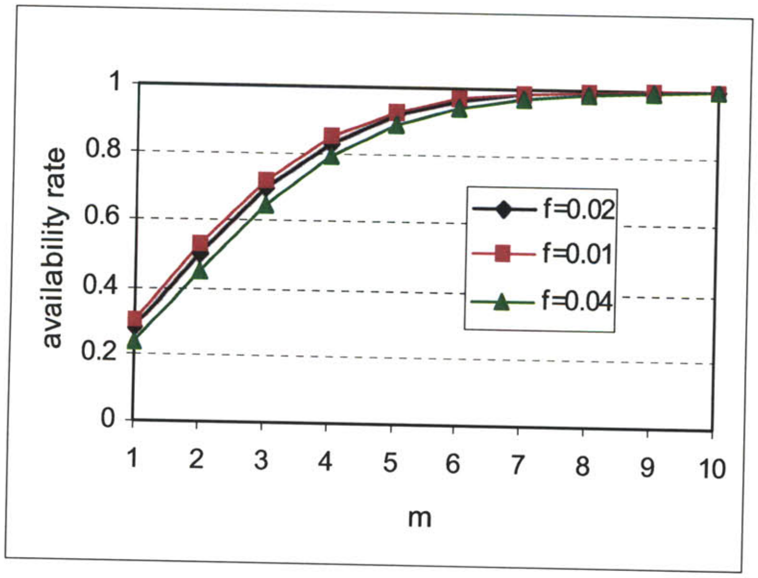


Figure 3.1: The availability rate with $p = 0.8$, $n = 10$, $N = 100$ and $m = 1 \sim 10$.

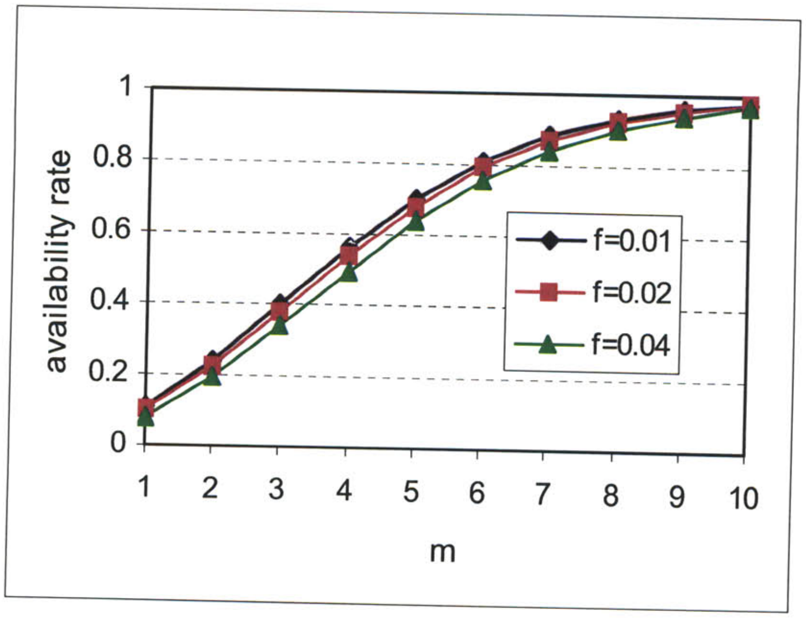


Figure 3.2: The availability rate with $p = 0.7$, $n = 10$, $N = 100$ and $m = 1 \sim 10$.

Figure 3.3 shows the tradeoff between per-node upload capacity and the availability rate. In this example, we assume that each node dedicates 500 Mbytes for collaborative storage. As we can observe that when per-node upload capacity goes up, the availability rate for a stored data item goes down due to the decrement of the number of redundant fragments. For the same availability rate, as peer online probability goes up, per-node upload capacity also increases.

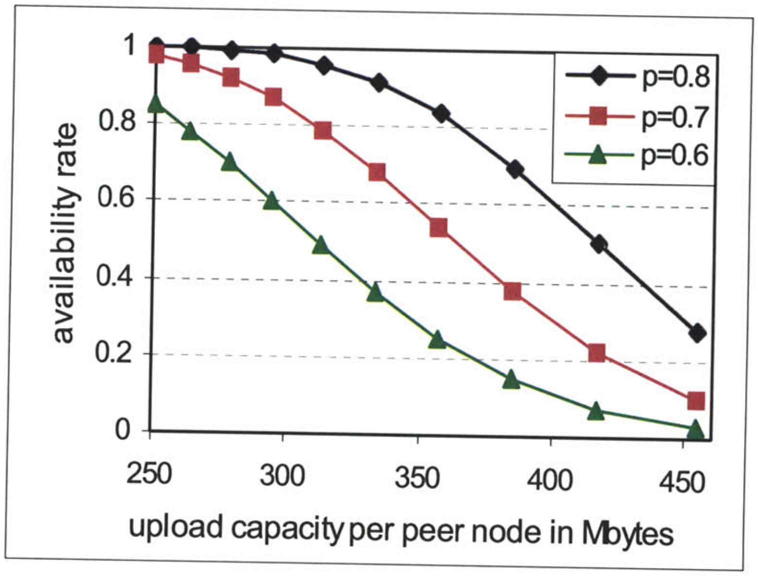


Figure 3.3: Tradeoff between upload capacity and availability rate with $f = 0.02$, $n = 10$, $N = 100$, and $m = 1 \sim 10$.

B. The file backup case:

It is reported that on average nearly 90% of the hard disk space is free for today's enterprise desktop PCs. Thus, there is tremendous economic incentive to use collaborative storage paradigm for file backups instead of using dedicated highly expensive backup servers/disks.

Regarding the availability rate for a stored data item using an erasure code with parameters of $(n, n+m)$ in the file backup case, we have the following closed-form formula:

$$Av\_rate = 1\text{-} f \times [1\text{-}(1\text{-} \sum_{i=0}^{m+n-1} (\binom{N}{i} \times (1-p)^{(N-i)} \times p^{i}))$$

$$\times(1-\sum_{j=0}^{n-1}(\binom{m+n}{j}((1-f)\times p)^{j}\times(1-(1-f)\times p)^{m+n-j}))]$$

where $2 \le n < N$, $1 \le m < N$, $m+n < N, 0 \le f, p \le 1$. (3.2)

If a stored file is not available, it has to be the case that both the original disk fails and the backup version stored on the peer nodes can not recover. The second part of the formula (the part starting with $f \times [1$-(1-...) indicates the quantity of the disk failure probability times the probability that the back up version stored on the peer nodes can not recover. The availability rate follows by subtracting the above quantity from one.

The corresponding numerical results are shown in Fig. 3.4~3.6. We see similar trends compared with the results in Fig. 3.1~3.3. However, compared with the results in Fig. 3.1~3.3 for the shared e-library case, we observe that the availability rate is slightly higher for file backup case with the same settings. This is due to the fact that a peer node itself has a local copy of the original data object and the data stored on other peer nodes are just for backup purpose.

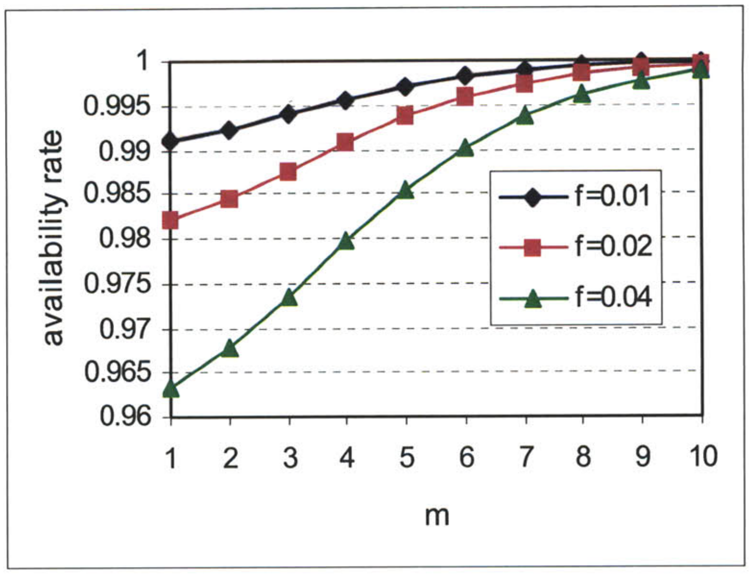


Figure 3.4: The availability rate for file backup case with $p = 0.7$, $n = 10$, $N = 100$ and $m = 1 \sim 10$.

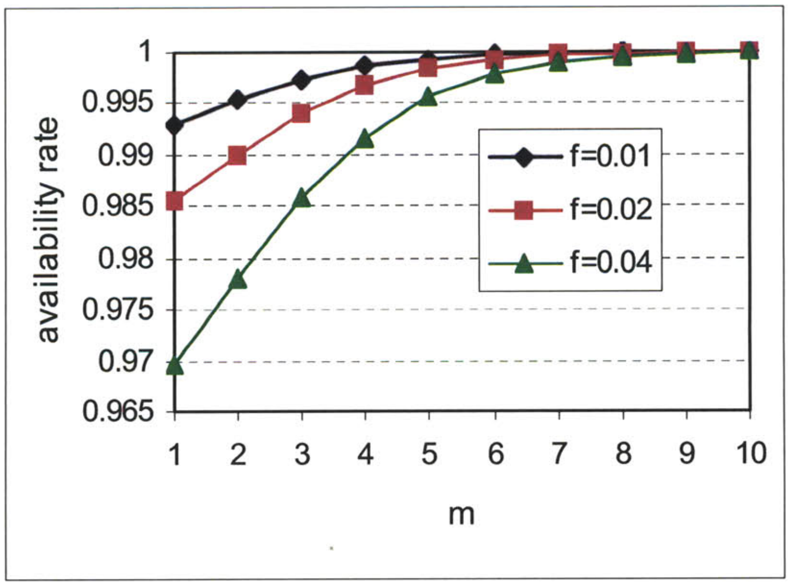


Figure 3.5: The availability rate for file backup case with $p = 0.8$, $n = 10$, $N = 100$ and $m = 1 \sim 10$.

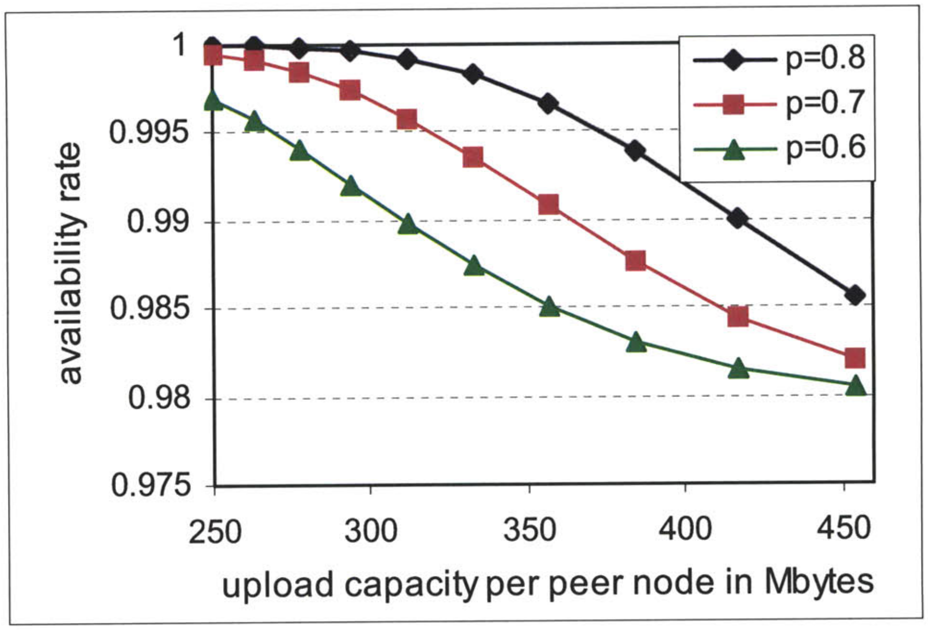


Figure 3.6: Tradeoff between upload capacity and availability rate for the file backup case with $f = 0.02$, $n = 10$, $N = 100$, and $m = 1 \sim 10$.

C. How many bad peers attribute to a really bad situation?

In this subsection, we intend to answer the following question: with the consideration of some *uncooperative* (bad) peers who might have deleted some fragments stored at their nodes for a given stored data item and have never shown up again, what the availability rate for that stored data item is going to be?

Let w denote the number of bad peers. For the shared e-library case, we have

$$Av\_rate = (1 - \sum_{i=0}^{m+n-1} (\binom{N}{i} \times (1-p)^{(N-i)} \times p^{i})) \times (1 -$$

$$\sum_{j=0}^{n-1} (\binom{m+n-w}{j} ((1-f) \times p)^{j} \times (1-(1-f) \times p)^{m+n-w-j}))$$

where $2 \leq n < N$, $1 \leq m < N$, $m+n < N$, $0 \leq w \leq m+n$, $0 \leq f, p \leq 1$. (3.3)

The first part of the formula, e.g., $(1 - \sum_{i=0}^{m+n-1} (\binom{N}{i} \times (1-p)^{(N-i)} \times p^i))$, means that we need to find $n+m$ peers from the total number of N peers to store the corresponding encoded fragments for a given data object in the first place, which is the same as the first part in Eq. 3.1. The second part of the formula, e.g., (1-$\sum_{j=0}^{n-1} (\binom{m+n-w}{j} (\ldots(1-(1-f)\times p)^{m+n-w-j}))$, means that after the storage of the $n+m$ encoded fragments on $n+m$ peer nodes (one per peer node), we need to retrieve n fragments out of $n+m-w$ encoded fragments after excluding w bad ones in order to recover the original data object. Essentially, Eq. 3.1 is a special case of Eq. 3.3 with the condition of w=0.

For the file backup case, we have

$$\text{Av\_rate}=1-f \times [1-(1 \sum_{i=0}^{m+n-1} (\binom{N}{i} \times (1-p)^{(N-i)} \times p^i)) \times (1-$$

$$\sum_{j=0}^{n-1} (\binom{m+n-w}{j} ((1-f)\times p)^j \times (1-(1-f)\times p)^{m+n-w-j}))]$$

where $2 \leq n < N$, $1 \leq m < N$, $m+n < N$, $0 \leq w \leq m+n$, $0 \leq f, p \leq 1$. (3.4)

Compared with Eq. 3.2, Eq. 3.4 means that after the storage of the $n+m$ encoded fragments on $n+m$ peer nodes (one per peer node), we need to retrieve n fragments out of $n+m-w$ encoded fragments after excluding w bad ones in order to recover the original data object. Essentially, Eq. 3.2 is a special case of Eq. 3.4 with the condition of $w=0$.

The corresponding numerical results are shown in Fig. 3.7 and Fig. 3.8. Fig. 3.7 shows the results with one bad peer for the shared e-library case with $p=0.8$, $n=10$, $N=100$ and $m=1\sim10$. Fig. 3.8 shows the results with one bad peer for the file backup case with the same settings as that in Fig. 3.7. Essentially, the curve is a shifted version to the right by the number of bad peers compared with the case without bad peers. Clearly, the more data redundancy, the more bad peers the system can tolerate. By comparing Fig. 3.7 and Fig. 3.8, we also observe that the impact by bad peers for the shared e-library case is more severe than that for the file backup case as a peer node in the file backup case has a local copy of the original data object on its own storage.

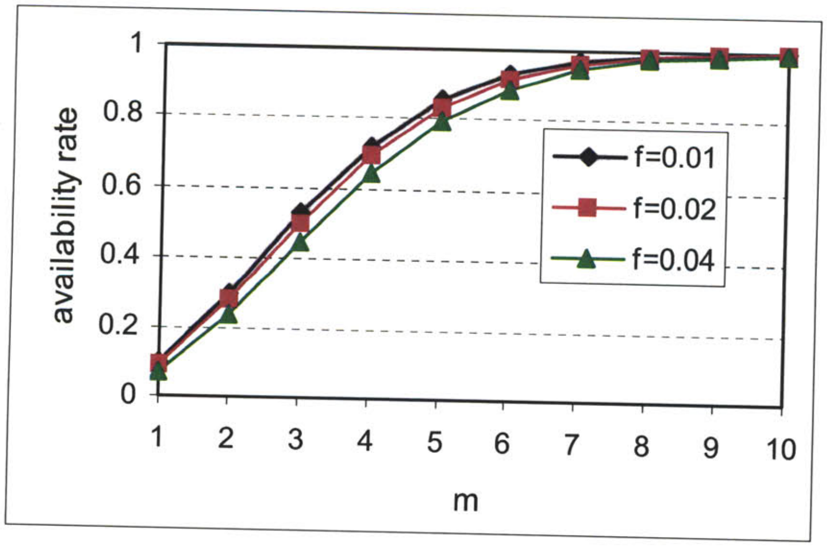


Figure 3.7: The availability rate for the shared e-library case with one bad peer and $p=0.8$, $n=10$, $N=100$ and $m=1\sim10$.

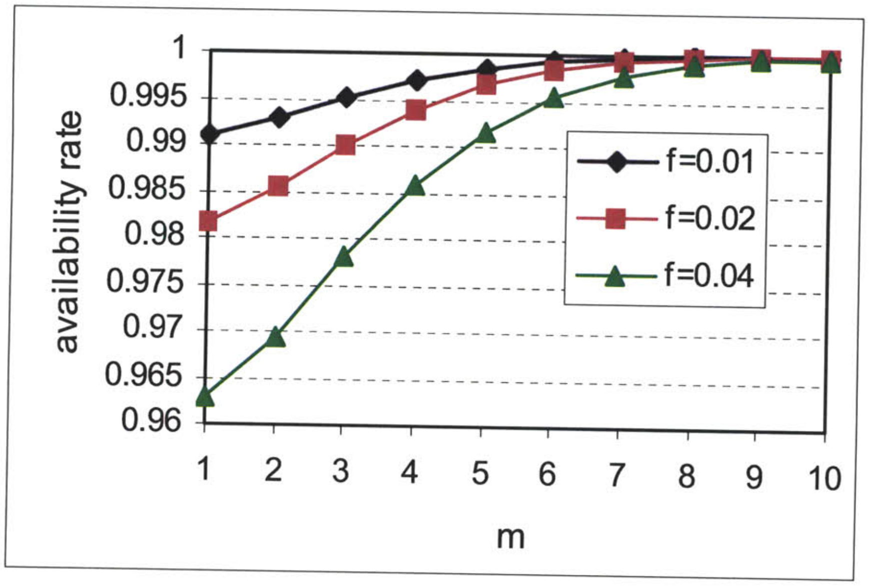


Figure 3.8: The availability rate for the file backup case with one bad peer and $p = 0.8$, $n = 10$, $N = 100$ and $m = 1 \sim 10$.

3.4.3.2 Scenario B: $n + m = N$

Let us take the shared e-library as an example and analyze how to maximize the availability rate of all e-books given limited storage space and predefined preference levels (dictated by the availability of the required number of peers in order to view an e-book) for each e-book. In this scenario, to view an e-book does *not* depend on which peers are available but only on how many peers are available because every peer has an encoded fragment for a given stored data object.

Depending on the importance of an e-book or how frequently it will be used, we may choose different erasure code parameters in the sense that the number of required peers' availability may vary for different e-books in order to view the given e-books. For example, one may only needs the availability of one other peer node in order to view an important e-book, in which case the erasure code only requires an encoded fragment from one other peer node and one local encoded fragment to recover the original data object. For an unimportant e-book, one may need to retrieve 10 encoded fragments from 10 other peer nodes in order to view the given e-book. The problem is how to maximize the availability rate of all e-books in the collaborative storage system given limited storage space and predefined preference levels for each e-book.

Let N be the total number of peers in the collaborative storage system and L be the total number of e-books to be stored on those peer nodes using collaborative storage paradigm. Let $d_i (1 \le i \le N)$ stand for the available storage space in bytes at i^{th} peer node for this joint effort of collaborative storage among those peers. Let $b_i (1 \le i \le L)$ indicate the size of the i^{th} e-book in bytes. In terms of the erasure code parameters, we assume that $n_i + m_i = N \ (1 \le i \le L)$ and n_i is always greater than one, where at least $(n_i - 1)$ peers' availability is required in order to view the i^{th} e-book (note that one encoded fragment is always available from the local node itself). The value of n_i is determined by the importance of the e-book for the peers. The more important of an e-book, the fewer peers' availability is required in order to view the given e-book. The problem is how to maximize the total availability rate of all the e-books given limited storage resources and predefined preference levels for each e-book. We further define $v_i (1 \le i \le L)$ as a variable to indicate the required number of peers' availability in order to view the i^{th} e-book. Another way to view this problem is how to minimize the required number of peers, e.g., $v_i (1 \le i \le L)$, in order to view the given e-books, given predefined preference levels, which are dictated by the set of parameters $\{n_1, n_2 \ldots n_i \ldots n_L\}$ that is given *a priori*. The optimization problem can be formulated as follows:

$$\text{minimize } \sum_{i=1}^{L} v_i \tag{3.5}$$

subject to:

$$1 \le v_i \le n_i - 1 \ , \ 1 \le i \le L \tag{3.6}$$

$$\sum_{i=1}^{L} \frac{b_i}{v_i + 1} \le d_j \ , \ 1 \le j \le N \tag{3.7}$$

Constraint (3.6) indicates the value range for each v_i. Constraint (3.7) ensures that the storage quota for the collaborative storage effort at each peer node is not surpassed. We will further explore this optimization problem with concrete numerical examples as one of our future directions.

We also assume that the online probability for each peer node is p. Without the consideration of disk failure rate the availability for the i^{th} e-book when a peer tries to view it is then given by

$$Av\_rate = 1 - \sum_{i=0}^{v_i - 1} \binom{N}{i} \times p^i \times (1-p)^{N-i} \tag{3.8}$$

The corresponding numerical results are shown in Fig. 3.9 and Fig. 3.10. Fig. 3.9 shows the results with $n + m = N$, $n = 1 \sim 10$ and $N = 20$. Fig. 3.10 shows the results with $n + m = N$, $n = 1 \sim 10$ and $N = 40$.

Compared with the case of $n+m<N$, this one has much higher availability rate due to much more redundancy in that each peer has stored an encoded fragment for a given data object. With the same setting, the bigger value of N, the higher is the availability rate. When N is reasonably large, e.g., N=40, even the peer online probability is very low, say $p=0.3$, the availability rate to view a given stored item is still very high with a reasonable data redundancy level.

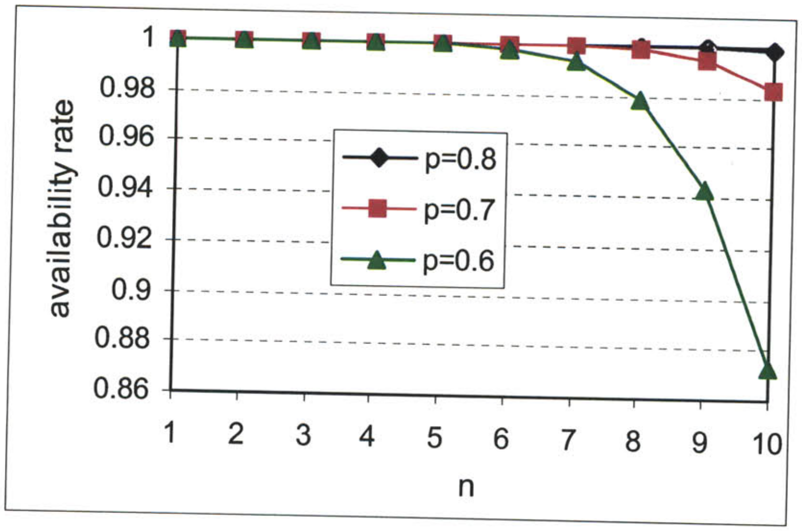


Figure 3.9: the availability rate for the share e-library case with $n+m=N$, $N=20$ and $n=1\sim10$.

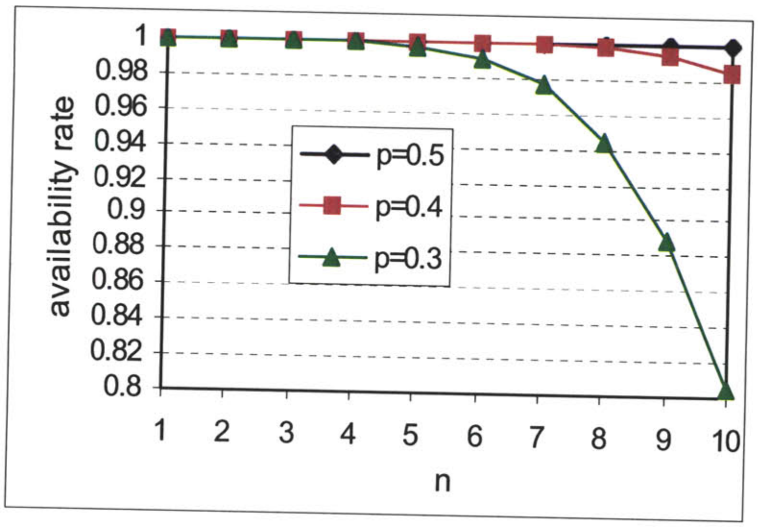


Figure 3.10: the availability rate for the share e-library case with $n+m=N$, $N=40$ and $n=1\sim10$.

3.5. Delay Analysis

We characterize the delay aspects of collaborative storage in P2P systems into two categories: the uploading delay and the retrieval delay. Let b stand for the size of a given data object in bytes and the erasure code parameter is $(n, n+m)$, where any n out of $(n+m)$ encoded fragments are required to recover the original data object. Let l_i $(1 \le i \le n+m)$ stand for the uploading link speed in bits per second for the i^{th} fragment between the uploading/retrieving node and the corresponding peer node to/from which the uploading/retrieving node is trying to upload/retrieve the i^{th} fragment (the i^{th} selected fragment in the retrieval case) of the given data object. Notably, the uploading delay consists of the preprocessing delay (mainly the encoding delay of the erasure code for the given data object), the transmission delay to transmit all $(n+m)$ encoded fragments as well as the corresponding propagation delay and queueing delay. On the other hand, the preprocessing delay for retrieval involves the decoding delay of the erasure code for the given data object. Let $d(n, n+m, b)_{enc}$ denote the processing delay for the erasure code to encode a data object of the size of b bytes with the parameters of $(n, n+m)$. Let $d(n, n+m, b)_{dec}$ stand for the processing delay for the erasure code to decode a data object of the size of b bytes with the parameters of $(n, n+m)$. Let $d_{p,i}$ indicate the propagation delay from the uploading/retrieving node to the peer node where the i^{th} fragment (the i^{th} selected fragment in the retrieval case) of the given data object will be uploaded/retrieved to/from. Let $d_{q,i}$ represent the queueing delay during the transmission of the i^{th} fragment (the i^{th} selected fragment in the retrieval case) of the given data object. Assuming sequential uploading, we have

$$d_{seq\_upld} = d(n, n+m, b)_{enc} + \sum_{i=1}^{n+m} \left(\frac{8 \times b/n}{l_i} + d_{p,i} + d_{q,i}\right) \tag{3.9}$$

Note that the first term is the preprocessing delay of the encoding of the given data object with erasure code, the second term is the sum of the transmission delay, propagation delay and queueing delay to sequentially upload each fragment to the corresponding peer node. The factor of 8 in the second term indicates the conversion from byte to bits.

Assuming parallel uploading, we have

$$d_{prl\_upld} = d(n, n+m, b)_{enc} + \max_{1 \le i \le n+m} \left(\frac{8 \times b/n}{l_i} + d_{p,i} + d_{q,i}\right) \tag{3.10}$$

Accordingly, the retrieval delay can be summarized as follows. Assuming sequential retrieval, we have

$$d_{seq\_rtvl} = d(n, n+m, b)_{dec} + \sum_{i=1}^{n}\left(\frac{8\times b/n}{l_i} + d_{p,i} + d_{q,i}\right) \tag{3.11}$$

Assuming parallel retrieval, we have

$$d_{prl\_rtvl} = d(n, n+m, b)_{dec} + \max_{1\le i\le n}\left(\frac{8\times b/n}{l_i} + d_{p,i} + d_{q,i}\right) \tag{3.12}$$

Notably, in the retrieval process only n fragments are involved instead of $(n+m)$ fragments for the uploading process.

Additionally, we also need to consider the delay to open the e-book reader. For example, in our experiments we use Mozilla browser as an e-book reader to open html files of those e-books. Sometimes, it takes a short while (in the order of several seconds) to open the Mozilla browser. It is desirable to use a light-weighted e-book reader instead of a full-featured web browser.

In general, the term of propagation delay is relatively small and queueing delay largely depends on the dynamic network traffic conditions. With the popularity of high-speed Internet and the continued trend of increased link speed (for example, the link speed for the new WiFi standard of 802.11n offers raw data rate up to 300 Mbps [155]), we expect that the preprocessing delay due to erasure code will be a dominant factor in a long run. Of course, when the file is very large, transmission delay is also noticeable.

Also, for collaborative offline Wikipedia, it may add one or more RTT (round trip time) among peer nodes to the total delay due to content search/inquiries within the entire Wikipedia.

As mentioned earlier, due to the huge size of the Wikipedia file with entire contents (2.9Gbytes in compressed form for the English version only by Dec. 2007), it is not feasible to encode/decode the whole Wikipedia file with Reed-Solomon code as it will be extremely slow due to the memory requirement and the processing overhead of the erasure code. In the following, we take collaborative offline Wikipedia as an example to analyze how to minimize the preprocessing delay imposed by this emerging application.

As mentioned in [148], we can use the bzip2recover tool (part of bzip2 distribution) to recover the individual parts of this compressed file. Basically, bzip2 splits its input into blocks of 900 Kbytes (by default), and compresses each of them individually. Bzip2recover seeks out the signature that identifies the beginning of each block, and dumps each block it finds as an individual file. With the help of the bzip2recover tool, we can convert the huge downloaded .bz2 file into a large set of smaller files, each of which is individually uncompressible. The problem is that the delay to encode/decode a 900 Kbytes file with Reed-Solomon code may be still large (see the experimental results in Fig. 3.11~3.17), depending on the erasure code parameters as well as the CPU speed of the peer node. As we will see from the experimental measurements in Section 3.6 regarding the encoding/decoding delay of Reed-Solomon code, the delay is still more than 10 seconds for a 900 Kbytes file with reasonable erasure code parameters on reasonably fast machines. Notably, bzip2 has nine options in terms of the block sizes when compressing a file, which are 100 Kbytes, 200 Kbytes, ..., 900 Kbytes (with the option of "-l" meaning the choice of 100 Kbytes block size and 900 Kbytes is the default block size). With this in mind, we could download the entire contents of Wikipedia file of the size of 2.9 Gbytes in compressed form (the block size is 900 Kbytes), uncompress it and recompress it by setting a reasonable block size for a reasonable preprocessing delay. We will encode the small files with Reed-Solomon code and spread the encoded fragments of each individual files among the peer nodes. When a file is requested, we first need to retrieve enough encoded fragments from peer nodes, decode it and recover it, then uncompress it.

In the following, we show how to optimize the block size of bzip2 for a reasonable preprocessing delay with Reed-Solomon code while the total number of small files is not too large in that too many small files add more complexity to calculate and maintain the index for content search within the entire Wikipedia.

Let N be the total number of peers in the collaborative storage system and W be the file size of the entire Wikipedia in bytes. The erasure code parameter is $(n, n+m)$, where any n out of $(n+m)$ encoded fragments suffice to recover the original data file. In terms of the erasure code parameters, we assume that every peer node has an encoded fragment for a given small file, i.e., $n+m=N$. Let l stand for the block size in Kbytes for bzip2 when compressing a file. Let $d(n,n+m,b)_{enc}$ and $d(n,n+m,b)_{dec}$ denote the encoding delay and decoding delay for the erasure code with parameters of $(n, n+m)$ for a file of b bytes respectively. Let F be the limit of the total number of small files. Let D denote the limit of the encoding/decoding delay. The optimization problem can be formulated as follows:

minimize $\quad l$ (3.13)

subject to:

$$l \in \{100, 200, \ldots, 900\} \tag{3.14}$$

$$\frac{W}{l \times 1024} \leq F \tag{3.15}$$

$$d(n, N, l \times 1024)_{enc} \leq D \tag{3.16}$$

$$d(n, N, l \times 1024)_{dec} \leq D \tag{3.17}$$

Constraint (14) indicates the value options for the variable l. Constraint (3.15) ensures that the total number of small files is limited. Constraint (3.16) and Constraint (3.17) means that the encoding /decoding delay to encode/decode the given small file of l Kbytes has to be smaller than D. We will further explore this optimization problem with concrete numerical examples as one of our future directions.

3.6. Experiments

We conduct two a series of quantitative measurements of the encoding/ decoding delay using Reed-Solomon code for a set of e-books with different erasure code parameters in a variety of circumstances. The large delay due to costly computation by Reed-Solomon code during the encoding and decoding process is one of the main reasons for the development of our fast and computationally much less-demanding erasure code, the community codes.

In the first part of the experiments, we repeated the same test for five times on the same machine and the results are very close for the same test with the same settings as environment is little changed on the same machine. In total, we conducted 600 runs of experiments in terms of encoding and decoding delay measurements. Our shared e-library prototype is written in Python and the precision of Python's time.clock() is in the order of microsecond. The Reed-Solomon codes implementation is based upon the open source code by Emin Martinian. The Reed-Solomon encoding/decoding is performed on three different types of machines with different computing capabilities. The three types of computers are: computers of relatively low-end with 1GHz CPU and 1 Gbytes memory; computers of middle-end with 1.8 GHz CPU and 2 Gbytes memory and computers of relatively high-end with 3.2 GHz CPU (dual core) and 3 Gbytes memory.

The sizes of the e-books used in the experiments are shown in Fig. 3.11, where the e-book sizes are in the order of Kbytes (ranging from 24 Kbytes to 704 Kbytes). The encoding and decoding delays in a variety of circumstances are shown in Fig. 3.12~3.17. We need to emphasize that the e-book sequence in the e-book size figure (Fig. 3.11) is *the same* as those appeared in Fig. 3.12~3.17 in terms of encoding/decoding delay of those e-books.

The results with Reed-Solomon code of 3-out-of-10 are shown in Fig. 3.12~3.14. The results with Reed-Solomon code of 14-out-of-20 are shown in Fig. 3.15~3.17. In general, the bigger of the e-book size, the bigger is the encoding/decoding delay with Reed-Solomon code with the same settings. We also observe that the bigger value of m, the longer is the delay for encoding in that more computation is needed to calculate more redundant fragments. On the other hand, the bigger value of n, the longer is the delay for decoding in that more computation is needed to recover the original n data fragments in order to reconstruct the original data object. In general we observe that the encoding delay is slightly larger than the decoding delay counterpart for Reed-Solomon code in all the tests we have performed with a variety of settings.

We notice that the retrieval delay reported in [123] is much smaller than the results we are reporting here. The seemingly disparity in terms of processing delay is caused by the dramatically different test setup in that a 42-machine cluster is used for the reconstruction of the file object in the test setup in [123] while we use only one single machine to perform the encoding/decoding of Reed-Solomon codes. For the applications that we are concerned, the encoding/decoding processes are most likely to be performed on a single machine of a peer node.

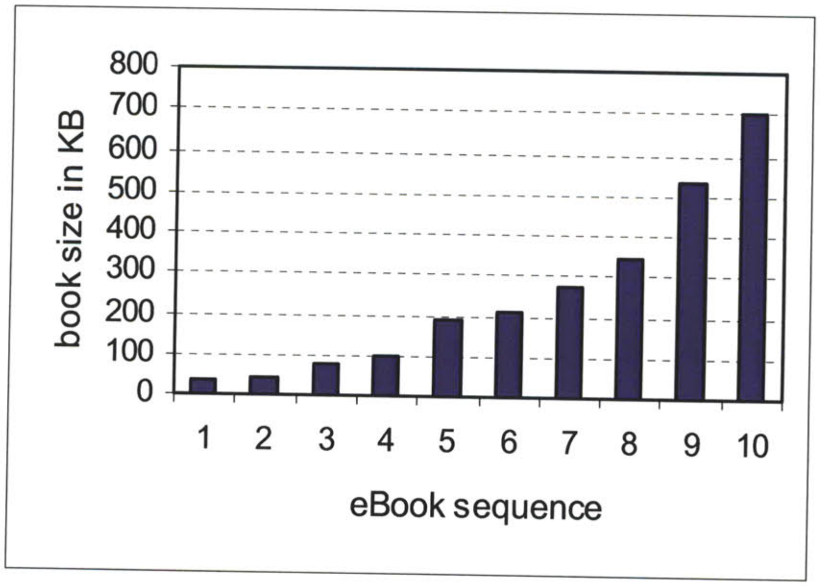


Figure 3.11: The sizes of the ten e-books used in the experiments.

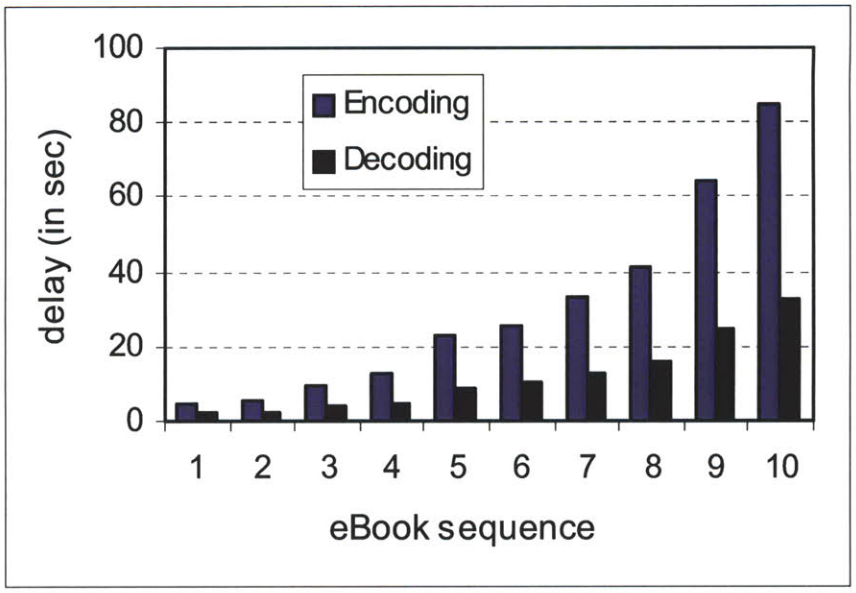


Figure 3.12: the encoding and decoding delay with $n = 3$, $n + m = 10$ on a computer with 1 GHz CPU and 1 Gbytes memory.

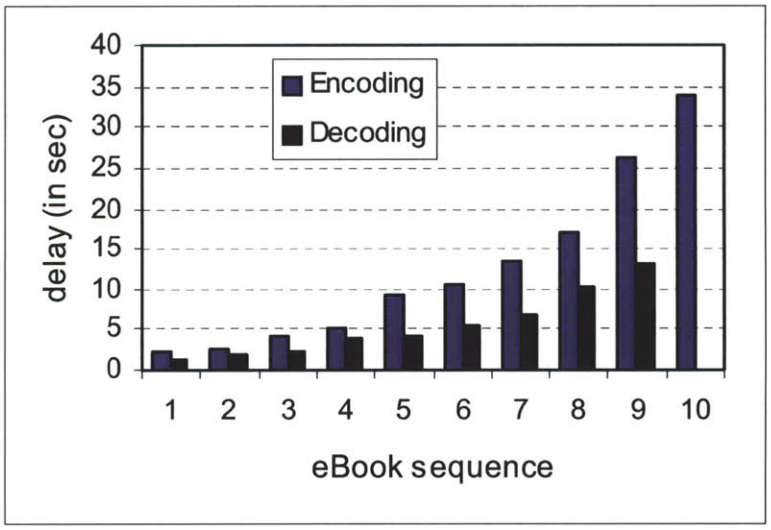


Figure 3.13: the encoding and decoding delay with $n = 3$, $n + m = 10$ on a computer with 1.8 GHz CPU and 2 Gbytes memory.

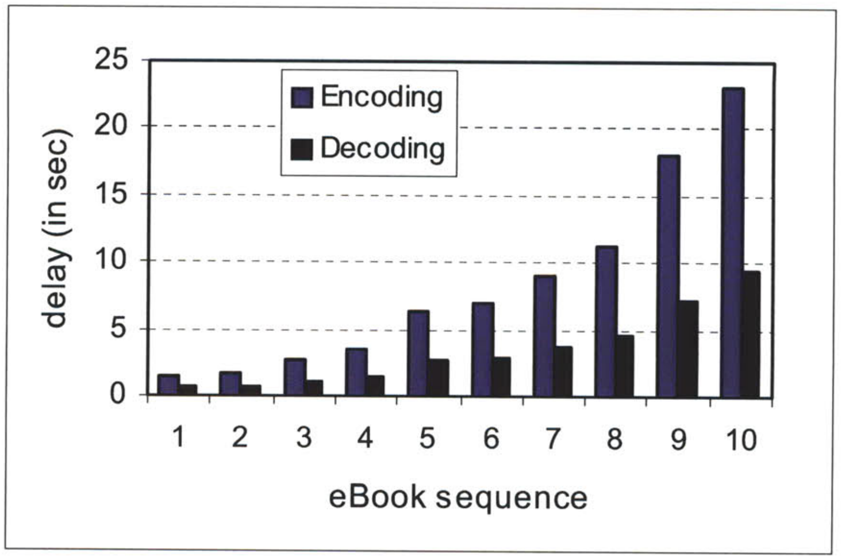


Figure 3.14: the encoding and decoding delay with $n=3$, $n+m=10$ on a computer with 3.2 GHz CPU and 3 Gbytes memory.

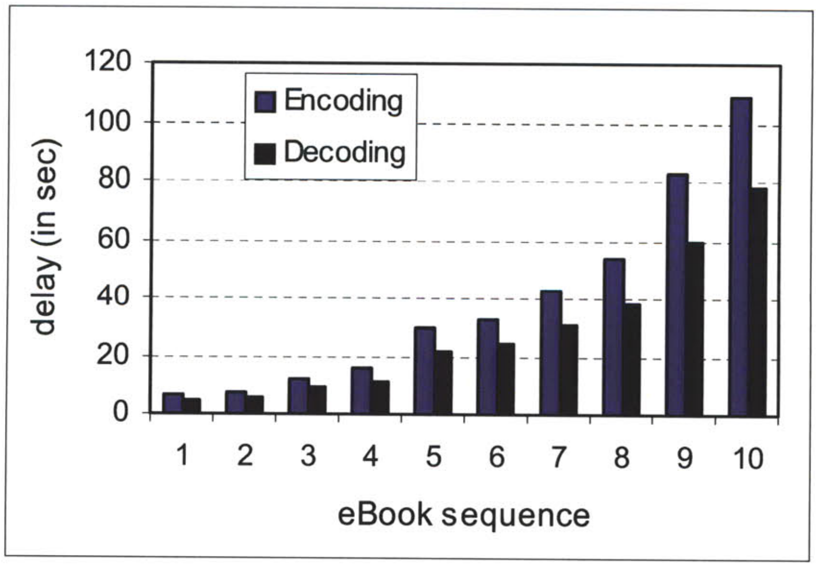


Figure 3.15: the encoding and decoding delay with $n=14$, $n+m=20$ on a computer with 1 GHz CPU and 1 Gbytes memory.

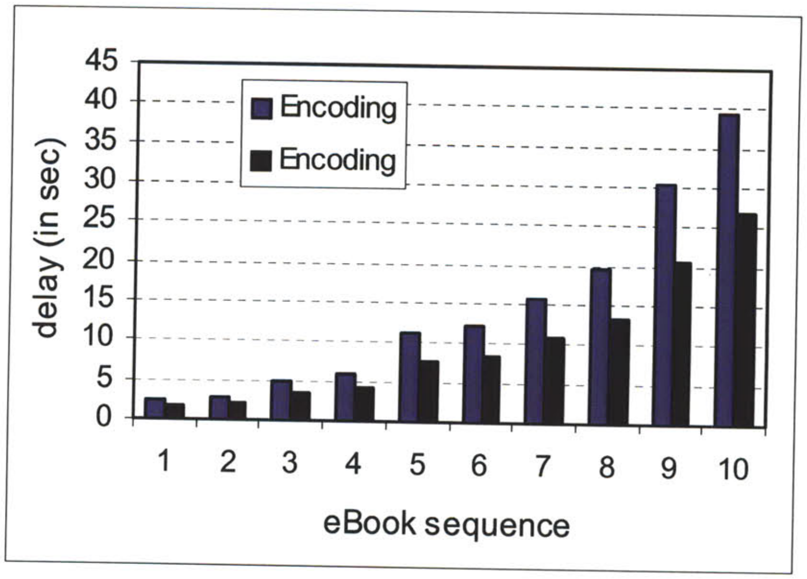


Figure 3.16: the encoding and decoding delay with $n = 14$, $n + m = 20$ on a computer with 1.8 GHz CPU and 2 Gbytes memory.

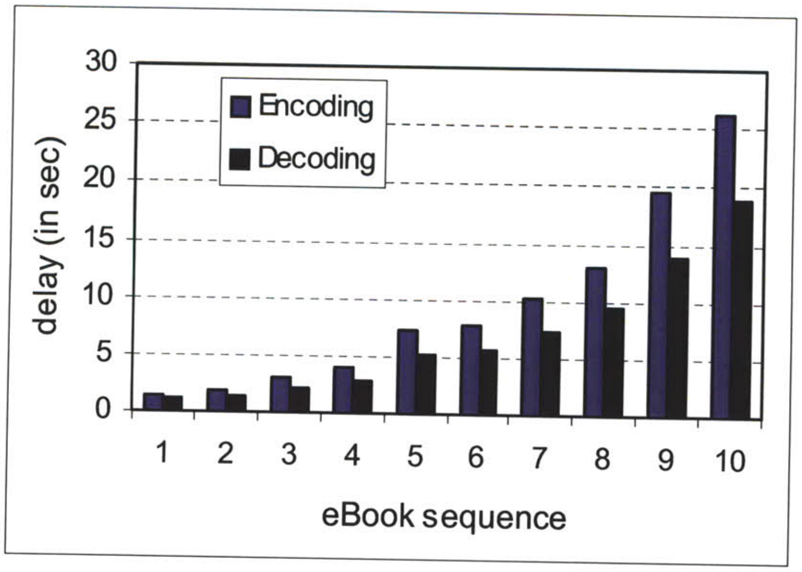


Figure 3.17: the encoding and decoding delay with $n = 14$, $n + m = 20$ on a computer with 3.2 GHz CPU and 3 Gbytes memory.

3.7. Prototypes

We build a shared e-library prototype based on the community storage paradigm with a group of network nodes.

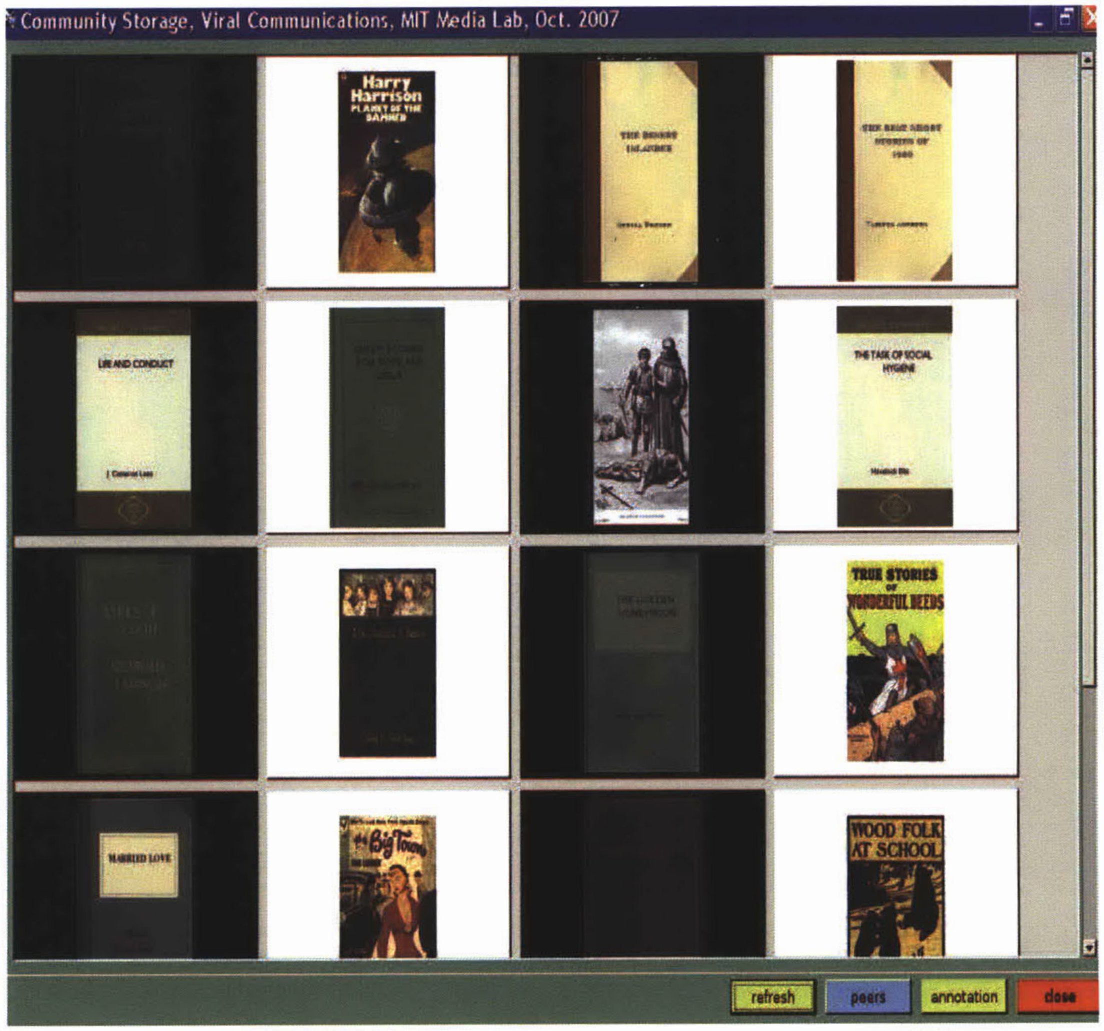
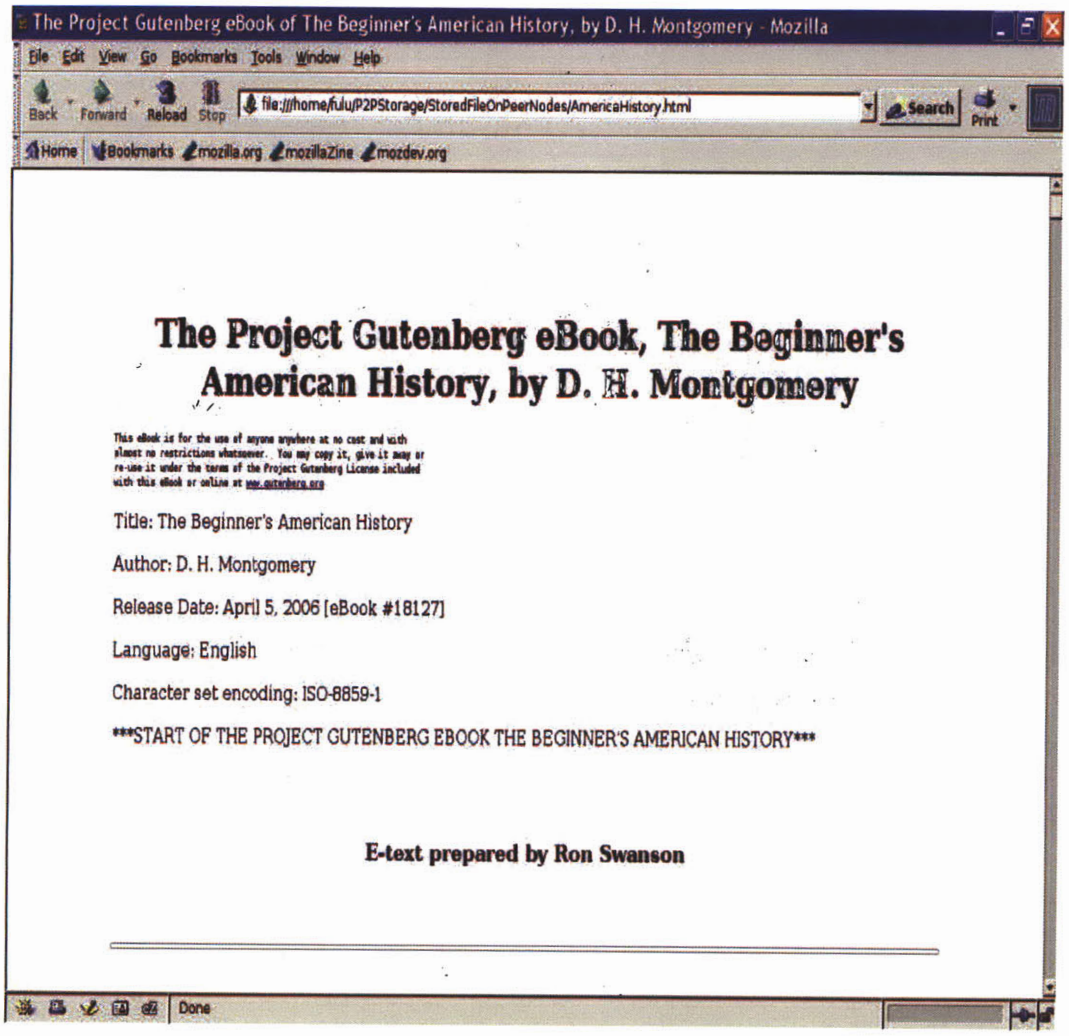

Community Storage, Viral Communications, MIT Media Lab, Oct. 2007
Harry Harrison
PLANET OF THE DAMNED
LIFE AND CONDUCT
THE TASK OF SOCIAL HYGIENE
TRUE STORIES OF WONDERFUL DEEDS
MARRIED LOVE
Big Town
WOOD FOLK AT SCHOOL
refresh
peers
annotation
close

Figure 3.18: the shared e-library prototype with a collection of e-books.

We conduct a series of qualitative experiments on the availability of e-books in the shared e-library prototype based on collaborative storage paradigm.

We repeatedly verify the availability of a set of e-books based on the collaborative storage paradigm by intentionally disconnect some of the peer nodes from the Internet (making it unavailable). We build a user interface of the prototype that one can browse the stored e-books with e-book cover image icons. For a given e-book, if there are enough peers online, i.e., one can retrieve enough encoded fragments to recover the original e-book, the e-book icon glows. On the other hand, if there are not enough peers online, i.e., one can not retrieve enough encoded fragments to recover the original e-book, the e-book icon grays out. We intentionally and selectively disconnect some of the peer nodes from the Internet, the availability of the e-books are always as what we expected. If we re-connect some of the peer nodes to the Internet, once there are enough encoded fragments for a given e-book that one can retrieve, the icon of the given e-book then turns into glowing again from graying out.


The Project Gutenberg eBook of The Beginner's American History, by D. H. Montgomery - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
file:///home/fulu/P2PStorage/StoredFileOnPeerNodes/AmericaHistory.html
Search Print
Home Bookmarks mozilla.org mozillaZine mozdev.org
Done

The Project Gutenberg eBook, The Beginner's American History, by D. H. Montgomery

This eBook is for the use of anyone anywhere at no cost and with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or online at www.gutenberg.org

Title: The Beginner's American History

Author: D. H. Montgomery

Release Date: April 5, 2006 [eBook #18127]

Language: English

Character set encoding: ISO-8859-1

START OF THE PROJECT GUTENBERG EBOOK THE BEGINNER'S AMERICAN HISTORY

E-text prepared by Ron Swanson

Figure 3.19: an e-book is reassembled and displayed.

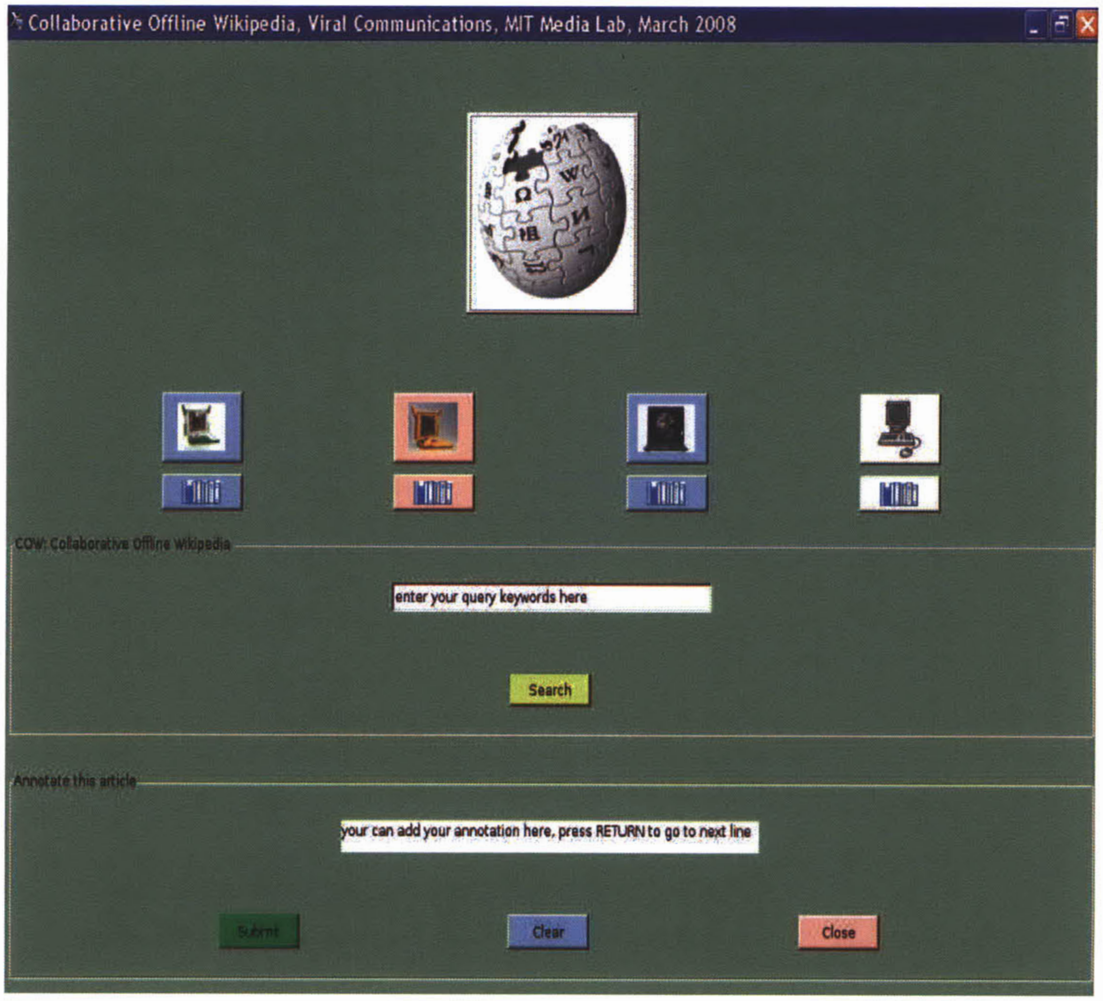

Collaborative Offline Wikipedia, Viral Communications, MIT Media Lab, March 2008
COW: Collaborative Offline Wikipedia
enter your query keywords here
Search
Annotate this article
your can add your annotation here, press RETURN to go to next line
Submit
Clear
Close

Figure 3.20: the collaborative offline Wikipedia prototype.

The community members are not only able to access all the contents of Wikipedia by providing a small portion of the total required storage even if they do not have the Internet access, they can also easily contribute articles and share annotations of the articles among the community.

In the following we show some empirical results in terms of the overhead (the retrieval delay in this example) when the number of required encoded blocks to reconstruct the original data object varies. In the example shown in Fig. 3.21, we assume that there is a data object of 1 Mbytes and the link speed is 3Mbps on average. The number of required encoded blocks to reconstruct the original data object ranges from 3 to 200. We assume that sequential retrieval is used for downloading the required blocks from other community member nodes. As we discussed earlier that the retrieval delay consists of the transmission delay, processing delay and queueing delay. For the sake of simplicity, we assume that the processing delay is in the form of (0.5 second + block size/Mbytes/sec) and the queueing delay ranges from 10 ms to 1 second.

Fig. 3.21 shows the retrieval delay for a varying number of required encoded blocks for a data object of 1 Mbytes. As we can observe from Fig. 3.21 that when the number of required encoded blocks increases, the overhead in terms of the retrieval delay also increases accordingly. Therefore, it is very important to consider a reasonable number of required encoded blocks when we use erasure code to encode the given data object to be stored on the community storage system. If the number of required encoded blocks is too small, the level of distribution and the level of load balance could be limited. If the number of required encoded blocks is too large, the overhead in terms of retrieval delay could be intolerable for the community members who use this system to upload and retrieve contents.

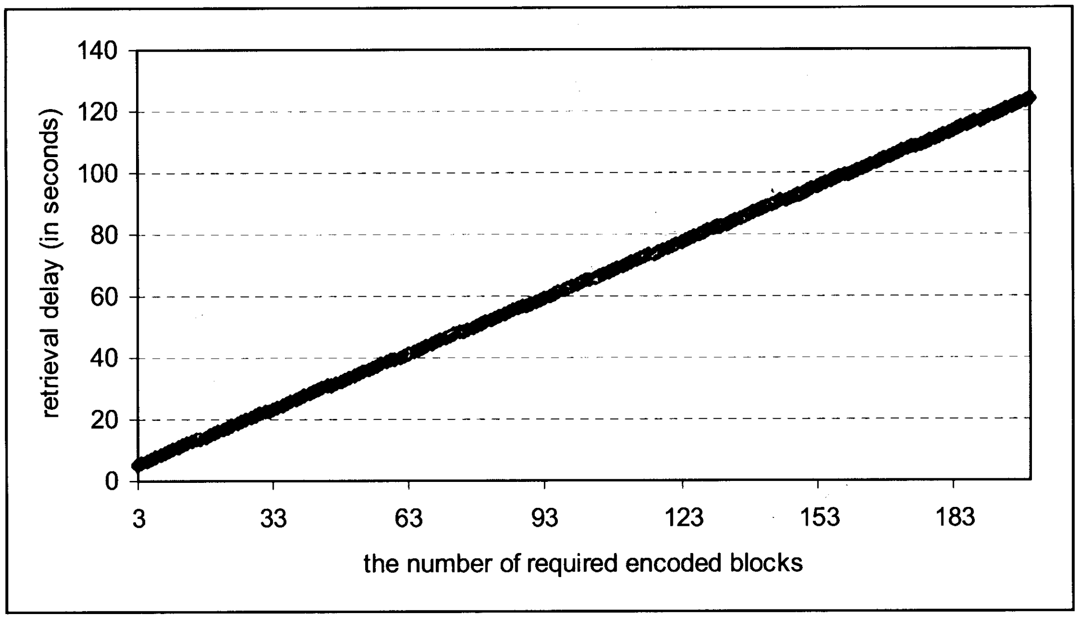


Figure 3.21: the retrieval delay for a varying number of required encoded blocks for a data object of 1 Mbytes.

3.8. Summary

In this chapter, we investigate the availability rate and uploading/retrieval delay in P2P collaborative storage systems, where contributory storage infrastructure is constructed among a group of peer nodes for the sake of high availability and storage efficiency. In particular, we focus on two emerging applications: the shared e-library with a collection of e-books and collaborative offline Wikipedia based on the collaborative storage paradigm among a group of peer nodes. For these two compelling applications, it is often burdensome to put the whole contents on one single node but collectively a group of peer nodes can hold the entire contents easily, each of which only needs to contribute a small amount of its own storage space. We take into account disk failure rate and peer online probability as well as erasure code parameters when deriving the availability rate for a stored data object in the collaborative storage system.

Due to the popularity of high-speed Internet and the continued trend of increased link speed, we believe that the dominant factor of uploading/retrieval delay will be the encoding/decoding delay imposed by erasure code. We conduct extensive experiments on the encoding /decoding delay of Reed-Solomon code in a variety of circumstances. As we can see from the experimental results that Reed-Solomon code must be performed with an input of a reasonably-sized data object/block in order to avoid substantial delays. For example, an ideal size for a data object/block with a reasonable Reed-Solomon code parameters (say 14-out-of-20) should be less than 200 Kbytes with a commodity computer (say 3.2 GHz CPU) for a tolerable delay (say 5 seconds for decoding).

The large delay due to costly computation by Reed-Solomon code during the encoding and decoding process is one of the main reasons for the development of our fast and computationally much less-demanding erasure code, the community codes.

Finally, we want to thank Emin Martinian for his open source code of the Reed-Solomon codes in Python.

Chapter 4

Community Sensing

The challenge here is how to aggregate the information sensed by various sensors around us to live in a better world as a community. It is reported that in 2008 there are more than 4 billion people in the world carrying a cell phone, typically with Bluetooth, cameras, etc. In particular, we are interested in the community computation aspects of a group of image sensors.

4.1 View Coverage

In this section, we study the probabilistic view coverage problem for image sensing with a community of image sensors in wireless sensor networks. The view coverage of an image sensor network determines the quality of the surveillance services that an image sensor network can provide. In this chapter, we present an indepth analysis on probabilistic view coverage in an image sensor network, where omnidirectional image sensors are randomly dropped to a given field and the locations of the image sensors may not be immediately known. We intend to answer the following question: if we randomly drop a given number of image sensors into a targeted field, what is the probability that a given area of interest can be effectively imaged and view-covered. The key to our analytical approach is to cast the probabilistic view coverage problem in wireless image sensor networks as a geometric one and then use the geometric techniques to find the solution. The analysis in this chapter provides probabilistic assurance of the view coverage that one can expect for random dropping of omnidirectional image sensors into a given field.

4.1.1. Introduction

As image sensing emerged as a hot research topic, the view coverage problem of an image sensor network with random dropping of image sensors, one of the fundamental issues for image sensor networks, has yet to be well explored. The view coverage of an image sensor network determines the quality of the monitoring services that an image sensor network can provide. With the advent of omnidirectional image sensors [157], image sensing has enabled a wide range of new applications such as object identification, localization and detection, etc.

Existing work on coverage problems in wireless sensor networks focuses on the sensing coverage issues. The seminal work by Meguerdichian et al in [106] addressed the best sensing coverage problem in a sensor network using computational geometry techniques to find the best support path where the region of interest and the locations of the sensors are known *a priori*. Recent work in [10] studied a new type of sensing coverage problem in wireless sensor networks, termed as barrier coverage, where the sensors form a barrier for the intruders in a given belt region. In [10], Balister et al further derived the critical density needed to achieve barrier coverage and/or connectivity in such thin strips of finite length.

While the sensing coverage problems [10,106] in wireless sensor networks have all been *well* explored, little has been known for the scenarios of probabilistic view coverage on some s*trategic spots* within a given region such as boundaries and corners with random dropping of omnidirectional image sensors. In a hostile environment, e.g., battlefields, hazardous areas, forests, mountains, etc., where sensors can not be deployed manually, random dropping of sensors to a given targeted area is required and the locations of the sensors may not be known. In this chapter we focus on the probabilistic view coverage of an image sensor network with random dropping of omnidirectional image sensors.

4.1.2. The Related Work

In the seminal sensing coverage work by Meguerdichian et al [106], the authors presented an optimal polynomial time algorithms that uses Voronoi diagram and Delaunay triangulation techniques to solve the best support path and the maximal breach path problems across a given sensor field where the locations of the sensors are known *a priori*. In [96], Liu et al showed that mobility improves sensing coverage of wireless sensor networks. In [152], Wang et al examined the sensing coverage problem in hybrid networks with both static and mobile sensors. They addressed the trade-offs between mobility and density for sensing coverage in wireless sensor networks. In [131], the authors considered a grid-based sensor network and they derived necessary and efficient conditions for the grid network to cover a given region.

The work in this section is partly motivated by the ringtoss game described in [83] by Larsen and Marx. Players throw a ring onto a grid of squares. If the ring touches no lines, then the player wins a prize. In [56], the authors analyzed the redundancy of sensing areas in a wireless sensor network to achieve energy efficiency by minimizing the redundant sensing areas covered by neighboring sensors.

As we address the problem of probabilistic view coverage of omnidirectional image sensors, to some extent the work in this chapter is also related to the packing of spheres [52,138]. As described in [138], the densest packing of identical circles in two dimensions is the hexagonal lattice packing (Appendix *E*), which is equal to 0.9069. It is known that optimal packing and covering in the plane are NP-complete [52].

4.1.3. Probabilistic View Coverage

In hostile and hazardous environments, e.g., battle fields or areas with biochemical hazards, image sensors have to be dropped into the fields of interest *remotely* and *randomly* and the locations of the dropped sensors may not be immediately known as the hostile environment may hinder the arrangement of communications. To understand the quality of surveillance in the above situations, we need to know the probabilistic assurance of the view coverage for a given area of interest that one can expect, in particular for some strategic spots like boundaries and corners.

4.1.3.1. The Model

An omnidirectional image sensor allows everything around the image sensor, i.e., full-circle 360 degree, to be imaged within a range [157]. Without loss of generality, we assume each omnidirectional image sensor can effectively detect an object of a given size, say s, within some distance from it, which is termed as the *effective view coverage radius* throughout this chapter. We also assume that all image sensors in the network are homogeneous with the same image sensing radiuses. If an object of a given size is too far away, it could become a single pixel on the image, which does not contain enough information to identify the given object. Of course, the area of view of a given image sensor is always affected by the line of sight such as obstacles, topography and other conditions like the weather (fogs, raining, snow), etc. In the following analysis, we will consider flat terrain of a given area under normal weather conditions. The situation under extreme conditions can be approximated from that of normal ones. Recent work in [84] presents a framework for Bayesian analysis of light field projections, which could be used to infer view coverage of image sensors under extreme conditions. We determined this as one of our future directions.

4.1.3.2. View Coverage in a Disk Field

In the first scenario, we consider a circular field with a radius of R, where N homogeneous omnidirectional image sensors are to be randomly deployed. Each image sensor has an *effective view* coverage radius of r, meaning for an object of a given size, say s, it can be effectively detected by the image sensor within the circular area centered at the image sensor with a radius of r. We assume that an image sensor's effective view coverage area is far smaller than the disk field, i.e., $r << R$. Throughout this chapter, we always assume that an image sensor is a point in the plane and an image sensor's effective view coverage is a circular area centered at the image sensor with a radius of r in the plane. Regarding the image sensor's effective view coverage in the given disk field, we have the following observations.

Proposition 4.1.1: Define A as the event that all of the N randomly-dropped image sensors' effective view coverage falls within the circular disk field, and we have

$$\Pr\{A\} = (\frac{(R-r)^2}{R^2})^N \tag{4.1}$$

As Illustrated in Fig. 4.1, if an image sensor's effective view coverage falls within the circular field with a radius of R, then the sensor has to fall within the shaded area, i.e., the inner circular area with a radius of $(R-r)$. The probability that a randomly-dropped sensor's view coverage falls within the disk field is the ratio of the shaded area of the inner circle and the area of the disk field, i.e., $\frac{\pi(R-r)^2}{\pi R^2} = \frac{(R-r)^2}{R^2}$. Thus if all of the N image sensors' effective view coverage falls within the disk field, the probability is $(\frac{(R-r)^2}{R^2})^N$, assuming that the dropping of each image sensor is *independent*. This concludes the proof of Proposition 4.1.1.

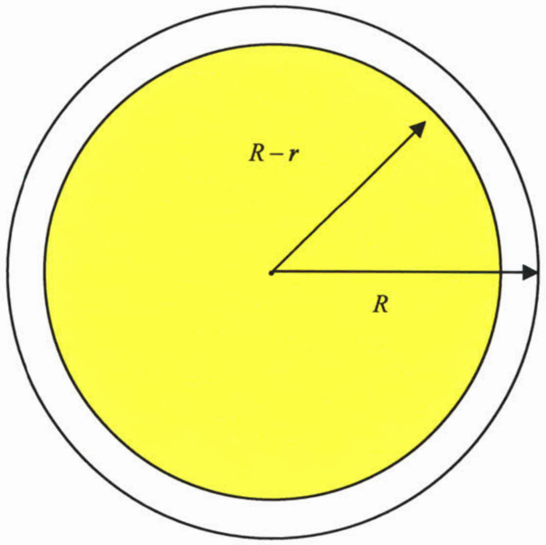


Figure 4.1: An illustration of the image sensor coverage.

Now let us define that event A_i stand for the case that the i^{th} image sensor's effective view coverage falls within the disk field without overlapping with other sensors' coverage.

Lemma 4.1.2: Regarding the probability that all of the N image sensors' effective view coverage fall within the disk field without overlapping among one another, we have

$$\Pr(A_1 \cap A_2 \cap \ldots \cap A_N\} \le \prod_{i=1}^{N} \frac{(R-r)^2 - (i-1)r^2}{R^2} \qquad (4.2)$$

To see this, we have

$$\begin{aligned}
&\Pr\{A_1 \cap A_2 \cap A_3 \cap \ldots \cap A_N\} \\
&= \Pr\{A_N \mid A_1 \cap A_2 \ldots \cap A_{N-1}\} \times \ldots \times \Pr\{A_2 \mid A_1\} \times P\{A_1\} \\
&= \Pr\{A_N \mid A_1 \cap A_2 \cap \ldots \cap A_{N-1}\} \times \ldots \times \Pr\{A_2 \mid A_1\} \times \frac{\pi(R-r)^2}{\pi R^2} \\
&\le \Pr\{A_N \mid A_1 \cap A_2 \cap \ldots \cap A_{N-1}\} \times \ldots \times \frac{\pi(R-r)^2 - \pi r^2}{\pi R^2} \times \frac{\pi(R-r)^2}{\pi R^2} \\
&\le \frac{\pi(R-r)^2 - \pi(N-1)r^2}{\pi R^2} \times \ldots \times \frac{\pi(R-r)^2 - \pi r^2}{\pi R^2} \times \frac{\pi(R-r)^2}{\pi R^2} \\
&\le \prod_{i=1}^{N} \frac{(R-r)^2 - (i-1)r^2}{R^2}
\end{aligned}$$

This concludes the proof of Lemma 4.1.2. Essentially, Lemma 4.1.2 provides an upper bound on the probability that all of the N image sensors' effective view coverage fall within the disk field without overlapping among one another.

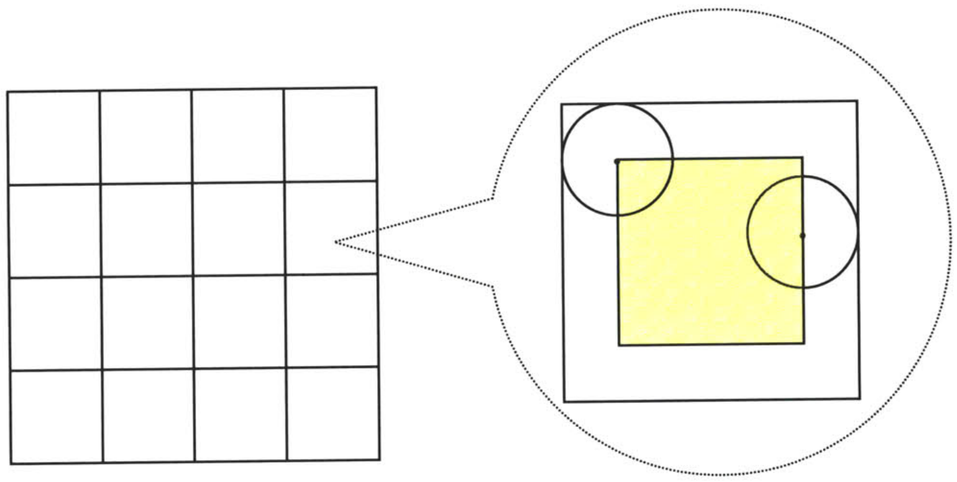
4.1.3.3. View Coverage in a Grid Field

Inspired by the ringtoss game described in [96], next we consider a rectangular $m \times n$ grid field (see Fig. 4.2.a), where the edges of the grid units consist of possible moving routes of targeted objects that may traverse this given area. Let l stand for the length of the edge of the small square grid unit and r denote the radius of the effective view coverage area by each omnidirectional image sensor. Assume N image sensors are randomly dropped to the given grid field.

Proposition 4.1.3: If $l \leq 2r$, the event that each image sensor's effective view coverage touches edges in the grid field is always true. Define H as the event that all of the N omnidirectional image sensors' effective view coverage touches no edges with $l > 2r$, and we have

$$\Pr\{ H \} = (\frac{l-2r}{l})^{2N} \tag{4.3}$$

The rationale behind Proposition 4.1.3 is illustrated in Fig. 4.2.b. If an image sensor's effective view coverage falls within the grid unit field with an edge length of l, then the sensor has to fall within the shaded area, i.e., the inner square area with an edge length of $(l-2r)$. The probability that a randomly-dropped image sensor's effective view coverage falls within a grid unit field is the ratio of the shaded area of the inner square and the area of the grid unit field, i.e., $\frac{(l-2r)^2}{l^2}$. Thus if all of the N image sensors' effective view coverage fall within grid unit fields without touching of the edges with $l > 2r$, the probability is $(\frac{(l-2r)^2}{l^2})^N = (\frac{l-2r}{l})^{2N}$, assuming that the dropping of each image sensor is *independent* from the droppings of other sensors. This concludes the proof of Proposition 4.1.3.



4.2.a: The grid field. 4.2.b: Sensor's view coverage in a grid unit.

Figure 4.2: The grid field and the randomly-dropped image sensors' effective view coverage.

Figure 4.3 shows different view coverage scenarios with different ratio values between the edge length of the grid unit field, i.e., l, and image sensor's effective view coverage radius, i.e., r, which will be used in the illustration of the following lemmas regarding the image sensors' effective view coverage of the conjunction points, i.e., the corners in the grid field, which may be regarded as strategic positions in the field. It might be of more importance for an image sensor to cover a corner in the field than part of a grid unit edge. In Fig. 4.3, the outer squares in each of the six sub-graphs depict the grid unit field and a ¼-circle is drawn with each of the four vertices of the grid unit field as its center to illustrate image sensors' effective view coverage scenarios.

In the following, we give a formal definition of the coverage symmetry rule (**CSR**).

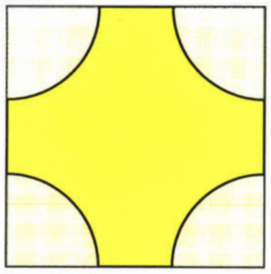
Definition: *If a circle centered at point X with a radius of r can cover a point Y, then a circle centered at point Y with the same radius of r must be able to cover point X.*

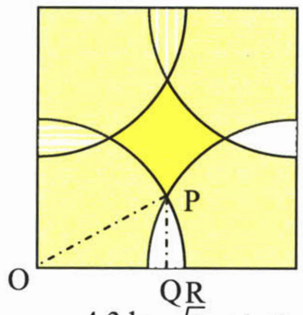
Without loss of generality, we call it the **coverage symmetry rule**. Throughout this chapter, we will repeatedly use this observation in the proof of some analytical results.

Proposition 4.1.4: Define H' as the event that all of the N omnidirectional image sensors' effective view coverage touches no corners in the grid field with $l > 2r$, we have

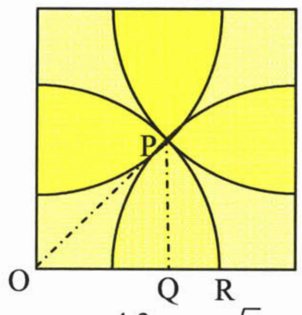
$$\Pr\{ H' \} = \left(\frac{l^2 - \pi r^2}{l^2}\right)^N \qquad (3.4)$$

As illustrated in Fig. 4.3.a, if an image sensor's effective view coverage touches no conjunction points, then the sensor has to fall within the central shaded area, i.e., the inner area in the grid unit field with the exclusion of four ¼-circles. This observation is obvious due to the coverage symmetry rule. The probability that a randomly-dropped sensor's effective view coverage touches no corners is the ratio of the inner shaded area and the area of the grid unit field, i.e., $\frac{l^2 - \pi r^2}{l^2}$. Thus if all of the N image sensors' effective view coverage touches no corners in the grid field with $l > 2r$, the probability is $\left(\frac{l^2 - \pi r^2}{l^2}\right)^N$, assuming that each image sensor is dropped *independently*. This concludes the proof of Proposition 4.1.4.

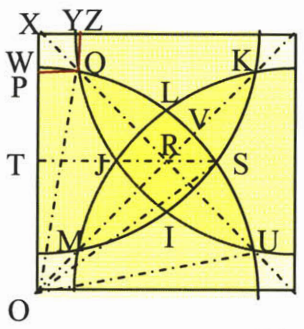

4.3.a: $l \geq 2r$



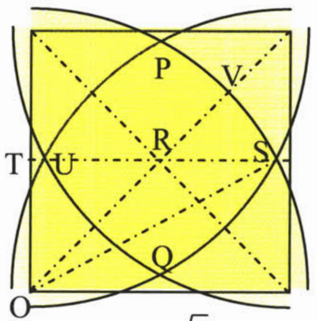
4.3.b: $\sqrt{2}r < l < 2r$



4.3.c: $l = \sqrt{2}r$



4.3.d: $r < l < \sqrt{2}r$



4.3e: $\frac{\sqrt{2}}{2}r < l \leq r$

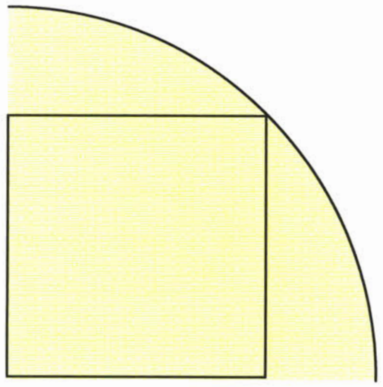

4.3.f: $l \leq \frac{\sqrt{2}}{2}r$

Figure 4.3: Image sensors' effective view coverage scenarios with different ratio values between l and r.

Lemma 4.1.5: Define E as the event that all of the N omnidirectional image sensors' effective view coverage touches no corners in the grid field with $\sqrt{2}r < l < 2r$, and we have

$$\Pr\{E\} = \left(\frac{l^2 - \pi r^2 + 4r^2 \arccos(l/2r) - 2lr\sin(\arccos(l/2r))}{l^2}\right)^N \quad (4.5)$$

As illustrated in Fig. 4.3.b, if an image sensor's effective view coverage touches no corners, then the image sensor has to fall within the central shaded area in the grid unit field. The probability that a randomly-dropped image sensor's effective view coverage touches no corners is the ratio of the central shaded area and the area of the grid unit field. Based on trigonometry, the ratio of the central shaded area and the area of the grid unit field is calculated as $\frac{l^2 - \pi r^2 + 4r^2 \arccos(l/2r) - 2lr\sin(\arccos(l/2r))}{l^2}$.

Thus if all of the N image sensors' effective view coverage touches no corners in the grid field with $\sqrt{2}r < l < 2r$, the probability is $\left(\frac{l^2 - \pi r^2 + 4r^2 \arccos(l/2r) - 2lr\sin(\arccos(l/2r))}{l^2}\right)^N$, assuming that each image sensor is dropped independently. This concludes the proof of Lemma 4.1.5.

Lemma 4.1.6: Define G as the event that each of the N omnidirectional image sensors' effective view coverage touches two corners in the grid field with $\sqrt{2}r < l < 2r$, and we have

$$\Pr\{G\} = \left(\frac{4r^2 \arccos(l/2r) - 2lr\sin(\arccos(l/2r))}{l^2}\right)^N \quad (4.6)$$

As illustrated in Fig. 4.3.b, if an image sensor's effective view coverage touches two corners, then the sensor has to fall within any of the four leaf-like overlapping areas by the ¼-circles centered at each of the four vertices of the grid unit field. The probability that a randomly-dropped image sensor's effective view coverage touches two corners is the ratio of four leaf-like overlapping areas and the area of the lattice unit field. The sum of the four leaf-like overlapping areas in Fig. 4.3.b is $8(\pi r^2 \arccos(\frac{l}{2r})/2\pi - l/2 \times r\sin(\arccos(\frac{l}{2r}))/2)$. The ratio of the four leaf-like overlapping areas and the area of the lattice unit field is equal to $\frac{4r^2\arccos(l/2r) - 2lr\sin(\arccos(l/2r))}{l^2}$. Thus if all of the N image sensors' effective view coverage touches two corners in the grid field with $\sqrt{2}r < l < 2r$, the probability is $(\frac{4r^2\arccos(l/2r) - 2lr\sin(\arccos(l/2r))}{l^2})^N$, assuming that each sensor is dropped independently. This concludes the proof of Lemma 4.1.6.

Lemma 4.1.7: With $l = \sqrt{2}r$, the event that each of the N omnidirectional image sensors' effective view coverage touches corners is always true. Define G' as the event that each of the N image sensors' effective view coverage touches two corners in the grid field with $l = \sqrt{2}r$, and we have

$$\Pr\{G'\} = (\frac{\pi}{2} - 1)^N \qquad (4.7)$$

As illustrated in Fig. 4.3.c, with $l = \sqrt{2}r$ the event that each of the N image sensors' effective view coverage touches corners is always true. This is due to the coverage symmetry rule (**CSR**). It is also shown in Fig. 4.3.c that if an image sensor's effective view coverage touches two corners, then the image sensor has to fall within any of the four petal-like shaded areas in the grid unit field. The probability that a randomly-dropped sensor's coverage touches two corners is the ratio of the four petal-like shaded areas and the area of the grid unit field. The ratio of the four petal-like shaded areas and the area of the grid unit field is equal to $\frac{\frac{\pi}{2}l^2 - l^2}{l^2} = \frac{\pi}{2} - 1$. Thus if each of the N image sensors' effective view coverage touches two corners in the grid field with $l = \sqrt{2}r$, the probability is $(\frac{\pi}{2} - 1)^N$, assuming that each image sensor is dropped independently. This concludes the proof of Lemma 4.1.7.

Corollary 4.1.8: Define $\hat{G}$ as the event that each of the N omnidirectional image sensors' effective view coverage touches one and only one corner in the grid field with $l = \sqrt{2}r$, and we have

$$\Pr\{\hat{G}\} = (2 - \frac{\pi}{2})^N \qquad (4.8)$$

As illustrated in Fig. 4.3.c and discussed in Lemma 4.1.7, with $l = \sqrt{2}r$ the event that a randomly-dropped image sensor to a grid field covers corners is always true. Further, a randomly-dropped omnidirectional image sensor to a grid field covers either one corner or two corners with $l = \sqrt{2}r$, depending on the landing areas in the lattice unit field. This is again due to the coverage symmetry rule as we discussed before. From Lemma 4.1.7, we obtain that the probability of the event that a randomly dropped image sensor's effective view coverage touches two corners in the grid field with $l = \sqrt{2}r$ is $(\frac{\pi}{2} - 1)$. Therefore, the probability of the event that a randomly dropped image sensor's effective view coverage touches one and only one corner in the grid field with $l = \sqrt{2}r$ is $1\text{-}(\frac{\pi}{2} - 1)$, i.e., $2 - \frac{\pi}{2}$. The result follows for N independently-dropped omnidirectional image sensors.

Lemma 4.1.9: Define $\overset{\bullet}{M}$ as the event that each of the N omnidirectional image sensors' effective view coverage touches one and only one corner in the grid field with $r < l < \sqrt{2}r$, and we have

$$\Pr\{ \overset{\bullet}{M} \} = 4^N (\frac{r}{l})^{2N} [\sin(\frac{\pi}{4} - \arccos(\frac{\sqrt{2}}{2} l / r))(1 + \cos(\frac{\pi}{4} - \arccos(\frac{\sqrt{2}}{2} l / r))) - \frac{\pi}{4} + \arccos(\frac{\sqrt{2}}{2} l / r)]^N \tag{4.9}$$

The proof of the lemma is given in Appendix *A*.

Lemma 4.1.10: Define $\hat{M}$ as the event that each of the N omnidirectional image sensors' effective view coverage touches exactly four corners in the grid field with $r < l < \sqrt{2}r$, and we have

$$\Pr\{ \hat{M} \} = (2 (2r^2 (\arccos(\frac{1}{2} l / r) - \frac{\pi}{4}) - l (r \sin(\arccos(\frac{1}{2} l / r)) - l / 2)) / l^2)^N \tag{4.10}$$

The proof of the lemma is given in Appendix *B*.

Lemma 4.1.11: Define $\dddot{M}$ as the event that each of the N omnidirectional image sensors' effective view coverage touches exactly three corners in the grid field with $r < l < \sqrt{2}r$, and we have

$$\Pr\{\dddot{M}\} = ((4(r^2 \arccos(\frac{\sqrt{2}}{2} l / r) - \frac{\sqrt{2}}{2} l \; r \sin(\arccos(\frac{\sqrt{2}}{2} l / r))) -$$

$$4r(2r(\arccos(\frac{1}{2} l / r) - \frac{\pi}{4}) - \sqrt{2} l \sin(\arccos(\frac{1}{2} l / r) - \frac{\pi}{4})))/l^2)^N \quad (4.11)$$

The proof of the lemma is given in Appendix *C*.

Corollary 4.1.12: Define $\ddot{M}$ as the event that each of the N omnidirectional image sensors' effective view coverage touches exactly two corners in the grid field with $r < l < \sqrt{2}r$, and we have

$$\Pr\{\ddot{M}\} = (1 - (\Pr\{\dot{M}\})^{1/N} - (\Pr\{\hat{M}\})^{1/N} -$$

$$(\Pr\{\dddot{M}\})^{1/N})^N \quad (4.12)$$

As illustrated in Fig. 4.3.d, the effective view coverage of a randomly dropped omnidirectional image sensor to the grid field with $r < l < \sqrt{2}r$ has four possible scenarios: (1) it covers one and only one corner (see Lemma 4.1.9); (2) it covers exactly two corners; (3) it covers exactly three corners (see Lemma 4.1.11); (4) it covers exactly four corners (see Lemma 4.1.10). From Lemma 4.1.9~4.1.11, the result follows for N independently-dropped image sensors.

Lemma 4.1.13: Define $\hat{W}$ as the event that that each of the N omnidirectional image sensors' effective view coverage touches more than four corners in the grid field with $\frac{\sqrt{2}}{2}r < l \leq r$, and we have

$$\Pr(\hat{W}) \geq ((4r^2(\arccos(\frac{l/2}{r}) - \frac{\pi}{4}) - 2l(r\sin(\arccos(\frac{l/2}{r})) - l/2))/l^2)^N \qquad (4.13)$$

The proof of the lemma is given in Appendix *D*.

In Fig. 4.3.f, we illustrate a ¼-circle centered at the left lower vertex of the grid unit field with $r \geq \sqrt{2}l$. As we can see that with $r >> l$ an omnidirectional image sensor's effective view coverage could touch many corners of the grid field and we do not discuss further due to limited space.

Theorem 4.1.14: There is a rectangular $m \times n$ grid field that consists of mn square units (see Fig. 4.2.a). Each randomly-dropped omnidirectional image sensor can cover one and only one square unit, assuming that the physical size of the sensor is much smaller than its effective view coverage area. In order to cover all of the mn square units in the grid field, the expected total number of randomly-dropped image sensors has to be $O(mn\ln(mn))$.

Proof: the expected number of randomly-dropped image sensors for the 1st covered square unit is 1. The expected number of randomly-dropped image sensors for the 2nd covered square unit is $\frac{mn}{mn-1}$ as the probability that a randomly-dropped image sensor covers a square unit other than the 1st covered square unit is $\frac{mn-1}{mn}$. Similarly, the expected number of randomly-dropped image sensors for the k^{th} covered square unit is $\frac{mn}{mn-(k-1)}$. So, in order to cover all the mn square units, the expected total number of randomly-dropped sensors (**ETS**) is

$$ETS = \sum_{i=1}^{mn} \frac{mn}{mn-(i-1)} \qquad (4.14)$$

We also have (see Appendix *F*)

$$\begin{aligned} &1+\frac{1}{2}+\ldots+\frac{1}{n} = 1\times 1+\frac{1}{2}\times 1+\ldots+\frac{1}{n}\times 1 \\ &\approx \int_{i=1}^{n+1} \frac{1}{x}dx = \ln x\,|_1^{n+1} = \ln(n+1)-\ln 1 = \ln(n+1) \approx \ln n \end{aligned} \qquad (4.15)$$

From (4.14), by bringing the term mn to the front we have

$$ETS = mn\sum_{i=1}^{mn} \frac{1}{mn-(i-1)} \qquad (4.16)$$

Define a new variable j to replace ($mn-(i-1)$) and Equation (4.16) can be rewritten as

$$ETS = mn\sum_{j=mn}^{1} \frac{1}{j} \qquad (4.17)$$

Equation (4.17) can be further rewritten as

$$ETS = mn\sum_{j=1}^{mn} \frac{1}{j} \qquad (4.18)$$

From Equation (4.15) and Equation (4.18), we have

$$ETS = mn\sum_{j=1}^{mn} \frac{1}{j} = O(mn\ln(mn)) \qquad (4.19)$$

This concludes the proof of Theorem 4.1.14.

4.1.4. Numerical Results

In Fig. 4.4, the number of randomly-dropped omnidirectional image sensors, i.e.,N, ranges from 1 to 10 in this illustration. From Fig. 4.4, we can see that the bigger value of the ratio between the radius of the circular field, i.e.,R and the radius of an image sensor's effective view coverage, i.e.,r, the more likely that all of the N randomly-dropped image sensors' effective view coverage fall within the circular disk field with the same value of N. On the other hand, the smaller of the value of the number of image sensors, i.e.,N, the more likely that all of the N randomly-dropped image sensors' effective view coverage fall within the circular disk field with the same ratio of R/r. When N is equal to one and the value of the ratio of R/r goes to infinity (R is far greater than r), the probability of the event A defined in Proposition 4.1.1 will asymptotically approach to one.

In Fig. 4.5, the number of randomly-dropped omnidirectional image sensors, i.e.,N, also ranges from 1 to 10. From Fig. 4.5, we can also see that the bigger value of the ratio between the length of the edge of the square grid unit field, i.e., l and the radius of an image sensor's effective view coverage, i.e.,r, the more likely that all of the N randomly-dropped image sensors' effective view coverage touches no edges in the grid field with $l > 2r$ for the same value of N. On the other hand, the smaller of the value of the number of image sensors, i.e.,N, the more likely that all of the N randomly-dropped image sensors' effective view coverage touches no edges in the grid field with $l > 2r$ for the same ratio of l/r. When N is equal to one and the value of the ratio of l/r goes to infinity (l is far greater than r), the probability of the event H defined in Proposition 4.1.3 will asymptotically approach to one.

Compared with Fig. 4.5 regarding the effective view coverage of the edges of the grid field, we see similar trends in Fig. 4.6 regarding the view coverage of the corners in the grid field. We observe that the bigger value of the ratio between the length of the edge of the square grid unit field, i.e., l and the radius of an image sensor's effective view coverage, i.e., r, the more likely that all of the N randomly-dropped image sensors' effective view coverage touches no corners in the grid field with $l > 2r$ for the same value of N. On the other hand, the smaller of the value of the number of image sensors, i.e., N, the more likely that all of the N randomly-dropped image sensors' effective view coverage touches no corners in the grid field with $l > 2r$ for the same ratio of l / r. When N is equal to one and the value of the ratio of l / r goes to infinity (l is far greater than r), the probability of the event H' defined in Proposition 4.1.4 will asymptotically approach to one.

Fig. 4.7 and Fig. 4.8 illustrate the trend of the probability of the event regarding the effective view coverage of the corners in the lattice field with $l = \sqrt{2}r$. Notably, with $l = \sqrt{2}r$ the probability that an image sensor's effective view coverage touches two corners is a constant, i.e., $\frac{\pi}{2} - 1$, independent of the value of l or r. Likewise, with $l = \sqrt{2}r$ the probability that an image sensor's effective view coverage touches one and only one corner is also a constant, i.e., $2 - \frac{\pi}{2}$, independent of the value of l or r. As we can observe from both Fig. 4.7 and Fig. 4.8 that the larger the value of the number of randomly-dropped sensors, i.e., N, the less likely that all of the N randomly-dropped image sensors' effective view coverage touches either two corners or only one corner. As N goes to infinity with $l = \sqrt{2}r$, the probability of the event G' defined in Lemma 4.1.7 and the event $\hat{G}$ defined in Corollary 4.1.8 will both asymptotically approach to zero.

Finally, we illustrate the trend of the probability of the event regarding the view coverage of all grid field units presented in Theorem 4.1.14 in Fig. 4.9. In this illustration, without loss of generality we consider that m is equal to n, i.e., square grid fields. In Theorem 4.1.14, we assume that each randomly-dropped sensor can cover one and only one lattice unit field. According to Theorem 4.1.14, in order to view-cover all of the m^2 grid field units, the expected total number of randomly-dropped omnidirectional image sensors becomes $m^2 \ln(m^2)$, i.e., $2m^2 \ln(m)$. As we can observe from Fig. 4.9 that with the growing of the number of grid field units, the expected total number of randomly-dropped image sensors to view-cover all the grid field units grows almost *linearly*, which is due to the counter effects of the square function of m^2, and the logarithm function of $\ln(m)$.

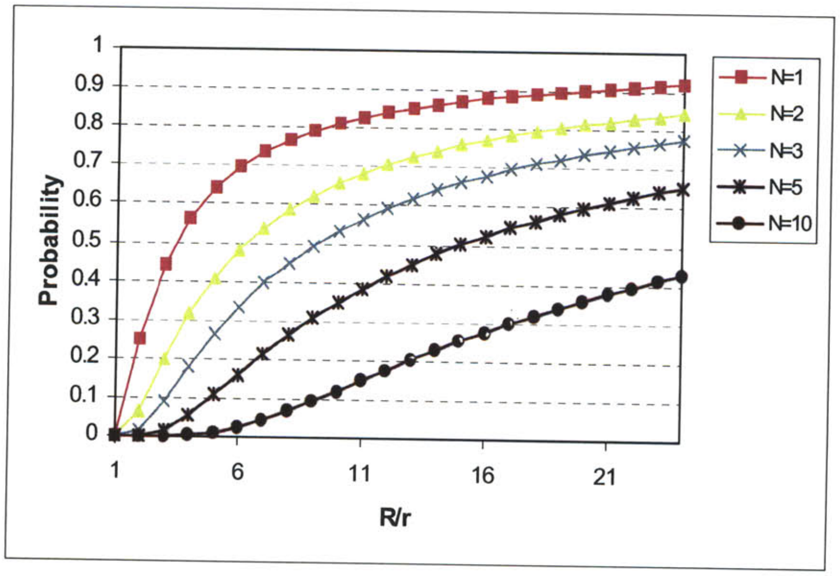


Figure 4.4: Numerical results for Proposition 4.1.1. The x-axis indicates the value of the ratio between R and r. The y-axis is the probability of event A defined in Proposition 4.1.1.

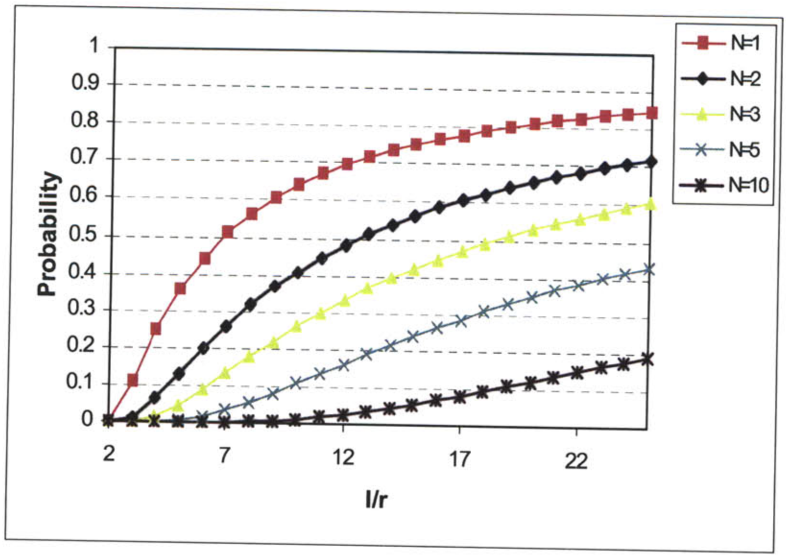


Figure 4.5: Numerical results for Proposition 4.1.3. The x-axis indicates the value of the ratio between l and r. The y-axis is the probability of event H defined in Proposition 4.1.3.

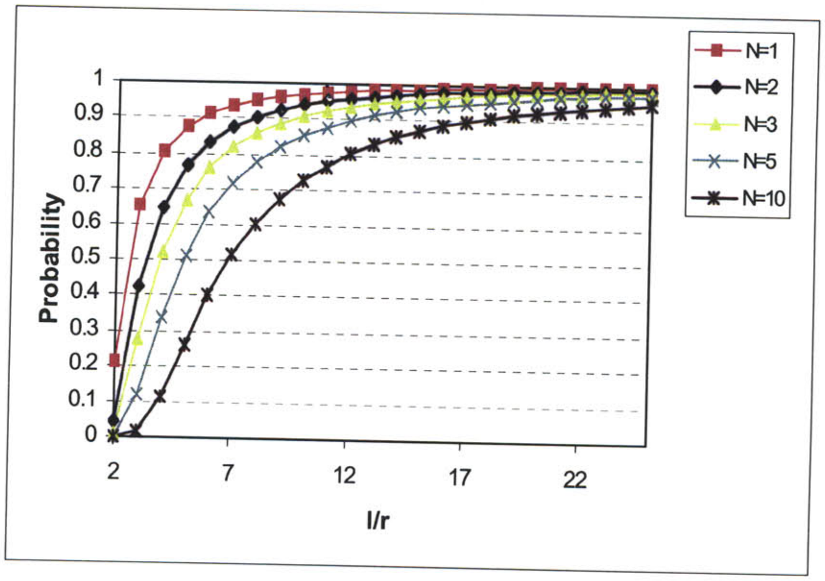


Figure 4.6: Numerical results for Proposition 4.1.4. The x-axis indicates the value of the ratio between l and r. The y-axis is the probability of event H' defined in Proposition 4.1.4.

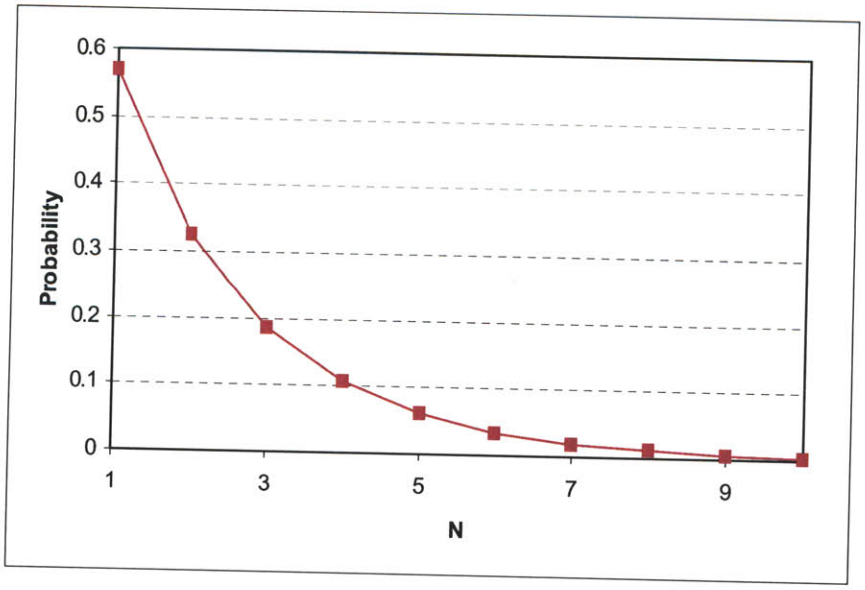


Figure 4.7: Numerical results for Lemma 4.1.7. The x-axis indicates the value of N. The y-axis is the probability of event G' defined in Lemma 4.1.7.

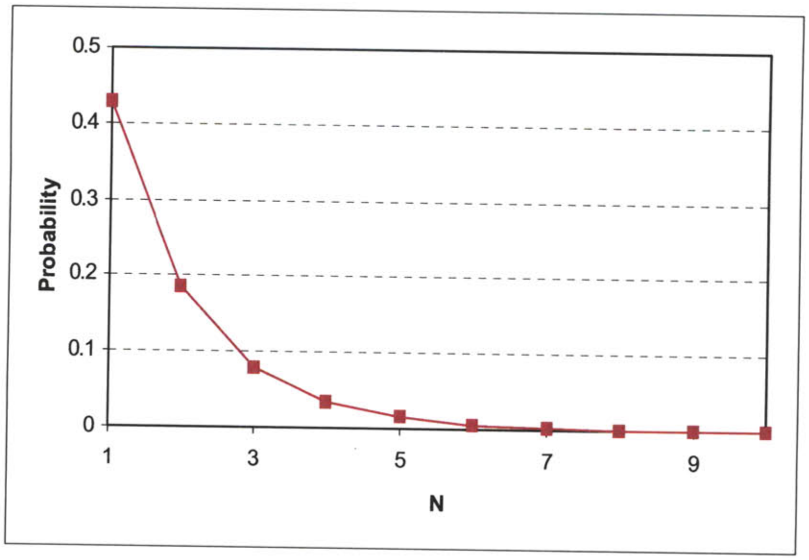


Figure 4.8: Numerical results for Corollary 4.1.8. The x-axis indicates the value of N. The y-axis is the probability of event $\hat{G}$ defined in Corollary 4.1.8.

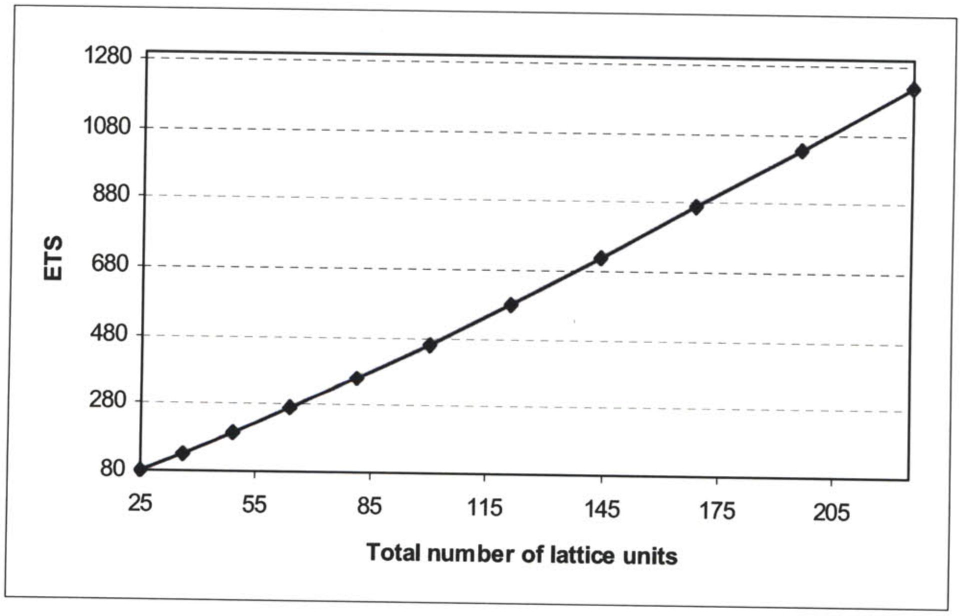


Figure 4.9: Numerical results for Theorem 4.1.14. The x-axis indicates the value of mn, i.e., the total number of grid field units. The y-axis is value of the expected total number of randomly-dropped omnidirectional image sensors (ETS).

4.1.5. Summary

In this section, we present an indepth analysis on the probabilistic view coverage problem in wireless sensor networks with random droppings of omnidirectional image sensors. We assume that each image sensor is randomly dropped to a targeted field and the locations of the sensors may not be immediately known, where hostile environments such as battlefields or hazardous circumstances may hinder the arrangement of communications. We derive the probability of view coverage assurance in a variety of circumstances using the coverage symmetry rule, geometric techniques and applied probability. We also derive that if a randomly dropped image sensor can cover one and only one grid unit field, the expected total number of randomly dropped image sensors has to be $O(mn\ln(mn))$ in order to cover all of the mn grid unit fields.

4.2. Image Data Aggregation

In this section we address the problem of efficient image data aggregation in community sensing. We present a novel framework of photo tourism [139] via wireless image sensing with a group of image sensors. We address several of the key challenges to achieve Photo tourism experience efficiently and effectively in this new platform. For example, we present efficient algorithms and protocols to minimize the overall communication loads during the image data aggregation process among images collected from a group of image sensors to reduce the redundancy before sending the aggregated image data out to the remote Photo tourism center. We also build image sensor prototypes with commodity cameras to demonstrate it. Essentially, we propose to conduct clustering and data compression at the sensor network level for the sake of reducing communication cost and energy consumption, instead of doing it at the Photo tourism stage.

4.2.1. Introduction

Photo tourism is developed by Microsoft and the University of Washington to transform regular digital photos into 3D and 360-degree experience [139]. On Aug. 20, 2008, Microsoft officially released Photo tourism to the public, allowing users to upload their images and create their own Photo tourism experience. The Photo tourism technology works in two steps [139]: the first step is to analyze multiple photographs taken of the same area; the second step is to intelligently display and navigate through the 3D *point cloud of features* identified in the first step. In this work, we envision a novel framework to create even "richer" Photo tourism experience from images *automatically* generated by wireless image sensors, even for some *humanly inaccessible* places.

With the advent of image sensors with wide field of view (180 ~ 360) [4,5,157], the advancement of image processing and computer vision techniques (image segmentation, motion estimation, etc.) [147], the improvement of low power image sensing technologies, wireless image sensing has enabled a wide range of new applications. Nowadays, image sensors are being used widely in defense, homeland security, entertainment, biomedical, education and scientific communities. In this chapter, we envision wireless image sensors with wide field of views, where *nearly* everything around the image sensor to be imaged within a range [4,5,157].

In this work, we present a novel framework to create even "richer" Photo tourism experience via wireless image sensing with a group of image sensors, in particular for some places like remote deserts, mountain pinnacles, hazards areas, even battleground fields, where image sensors can not be placed *manually*. In the above scenarios, image sensors have to be dropped to the given area *remotely* and *randomly*. Those image sensors *self-organize* themselves into a network to accomplish some tasks like location estimation, time synchronization, image data aggregation and image data transmission, etc., *cooperatively*. We address several key challenges in this new platform. We just list a few as follows:

- *How to aggregate the image data in such a way that the overall communication load among all image sensors is minimized while maximizing the number of common feature points among images from nearby image sensors during image data aggregation process?*

- *How to maximize the lifetime of the image sensor network given limited energy resources equipped at each image sensor node?*
- *How to construct the Photo tourism experience with images obtained from the image data aggregation process in a simple and efficient way?*

We present a distributed algorithm in tackling the first two challenges in two steps. The first step is to remove the redundant images before image data aggregation. The basic idea is that if one image sensor's field of view is already covered by images from its neighboring image sensors, the image from that sensor is no longer valuable from Photo tourism perspective at a later stage. The second step is to choose several *supersensors* among all image sensors in the network as local image data aggregation *leaders* based on multiple factors, which include the Euclidean distance among image sensors, the residual power at each image sensor, the network distance among image sensors, as well as the communication channel conditions among image sensors. In the image data aggregation process, a supersensor broadcasts the information of its own image data and all the *feature points*, say f_i s, in its image data to its cluster members to take advantage of the *wireless broadcast advantage* (*all* of the intended receivers within the communication range can get the data with *one single* transmission) and each cluster member will send back only the *difference* of the image data (the *shift* values of *feature points positions*, say, Δf_i s, if any in the *overlapping* area as well as the image data in the *non-overlapping* areas) to the corresponding supersensor for aggregation. In some sense, the problem of choosing several supersensors as local image data aggregation bases is related to the domain of *clustering* techniques [53], but none of them is directly applicable to the problem we are addressing, that is the minimization of the overall communication load among all image sensors while maximizing the number of *common feature points* during the image data aggregation process. Moreover, we need to take into account multiple factors as we mentioned earlier. To some extent, the minimization of communication load and the minimization of energy consumption are *coupled* in that more communication load often leads to more energy consumption at image sensor nodes.

For the third challenge, the choosing of supersensors as local cluster heads also facilitates the construction of the Photo tourism experience at the later stage. The images from the local cluster heads serve as the bases for the construction of Photo tourism experience. All we need to do is to reconstruct the images with *common feature points* to reset $(f_i + \Delta f_i)$s and feed all the images into the *interpolation* process by Photo tourism.

The major contribution of this work is the presentation of a novel framework for Photo tourism via wireless image sensing with a group of image sensors in order to achieve even "richer" and "up-to-date" experience with images taken and uploaded *automatically* and *remotely*, for some places that are even *humanly inaccessible*. As the old saying goes, the most beautiful scenes are often observed from some dangerous vantage points, which are usually *humanly inaccessible*. Imagine that if we can randomly drop some image sensors to some mountain pinnacles or remote deserts, this type of platform would enable us to get "fresh" and "up-to-date" 3D and 360-degree experience of the beautiful scenes observed at those vantage points right on our computer screen in the office.

4.2.2. Related Work

This work is mostly motivated by the newly released groundbreaking work of Photo tourism [139] and the advancement of lower power image sensing technologies [4,5,157]. Photo tourism was born of the seminal research by Snavely, Seitz and Szeliski and it was initially termed as “photo tourism”. The basic idea is to create a 3D photo tour from a collection of images taken from the same place. We refer interested readers to [139] for more implementation details of the Photo tourism technology.

In [4,5], Aliaga et al presented techniques to reconstruct 3D interactive walkthroughts with omnidirectional images. One key feature in the approaches in [4,5] is to exploit the multiple paths that can be followed between images to increase the number of feature correspondences between distant images.

In [80], Koizumi and Ishiguro presented *town digitizing* with omnidirectional images using an interpolation process termed *zooming stereo*, where a zooming ratio between a pair of omnidirectional images is used as a parameter to estimate the changes of visual appearance along lines connecting two omnidirectional images.

In the image acquisition process in both [4,5] and [80], a powerwheel or motorized cart is used to take images around an environment. In this chapter, we consider a scenario in which a group of image sensors are *randomly* and *remotely* dropped to a given field, in particular for some *humanly inaccessible* areas. In our case, those randomly dropped image sensors self-organize themselves into a wireless image sensor *network*, which *automatically* generates and uploads the images from that given field to a *faraway* Photo tourism center. The major challenge in our case is how to self-organize those image sensors for efficient image data *aggregation* and *transmission*, as well as for *easy* construction of Photo tourism experience with those images.

The work in [149] by Tumblin et al proposed an approach to measure the intensity *difference* between adjacent pixel pairs among photosensors within a camera instead of measuring direct pixel intensity. The benefit of their approach is that it achieves finer quantization and higher contrast, compared with the existing intensity-based cameras. In this chapter, we consider the image data aggregation of multiple image sensors (cameras), where each image sensor only sends the *difference* of the image data (the *shift* values of *feature points positions*, say, Δf_i s, if any in the *overlapping* area as well as the image data in the *non-overlapping* areas) to its corresponding supersensor. The goal here is to reduce the overall communication loads among all image sensors while maximizing the number of common feature points among images from nearby image sensors during the image data aggregation process.

The LEACH (Low-Energy Adaptive Clustering Hierarchy) protocol presented in [72] proposed methods to utilize randomized rotation of local cluster heads to distribute the energy consumption evenly among the sensors in the networks so that the lifetime of the sensor network can be maximized. In the LEACH protocol, the cluster heads are *randomly* picked and each sensor chooses its cluster head based on the minimum required communication energy. In this chapter, the supersensors are chosen to *minimize* the total communication load during the image data aggregation process while *maximizing* the number of *common feature points* among images from nearby image sensors.

In [49], Estrin et al proposed localized algorithms for coordination in wireless sensor networks, where a promotion timer is set to be *inversely proportional* to the sensor's remaining power. The cluster head is chosen locally when one's promotion timer expires first to declare its promotion. In [49], the number of cluster heads is *not* fixed and in our case the number of supersensors is pre-determined and fixed. In [49], the criteria to choose the cluster heads are solely based upon the residual power of the sensor nodes. In our case the residual energy is just one of the factors to be considered in choosing the supersensors.

Traditionally, we often compress the image data soon after sensing, which leads to the waste of valuable sensing resources. In [9,13,48,126], a compressive image sensing platform is proposed, in which the signal is sampled at a greatly *reduced rate*. We will address the problem of Photo tourism via wireless image sensing in the context of compressive image sensing as one of our future directions.

4.2.3. The Framework

4.2.3.1. The Model

An image sensor with a wide field of view allows *almost* everything around the image sensor to be imaged within a range [4,5,157]. Without loss of generality, we assume each image sensor can effectively detect an object of a given size, say s, within some distance from it, which is termed as the *effective view coverage radius* throughout this chapter. We also assume that all image sensors in the network are *homogeneous* with the same image sensing radiuses. If an object of a given size is too far away, it could become a single pixel on the image, which does not contain enough information to identify the given object. Of course, the field of view of a given image sensor is always affected by the line of sight such as obstacles, topography and other conditions like the weather (fogs, raining, snow), etc. In the following discussion, we will consider flat terrain of a given field under normal weather conditions. The situation under extreme conditions can be approximated from that of normal ones.

4.2.3.2. The Framework

Given a group of image sensors that are *randomly* deployed in a field and a *far-away* Photo tourism center, we want to create the Photo tourism experience from images taken from those wireless image sensors in an efficient way. First of all, we want to remove those images whose *feature points* are already covered by images from neighboring image sensors. Then, we choose a portion of the image sensors, say k of them, as *supersensors*, each of which acts as a local image data aggregator to further reduce the *redundancy* among images from its neighboring sensors while maximizing the number of *common feature points* before sending out to the remote Photo tourism center.

An illustration of the image data fusion model is given in Fig. 4.12. The local image data aggregation process operates as follows: *each supersensor broadcasts the information of its own image data and all the feature points, say f_i s, in its image data to its cluster members. Each cluster member will send back only the difference of the image data (the shift values of feature points positions, say, Δf_i s, if any in the overlapping area as well as* *the image data in the non-overlapping areas) to the corresponding supersensor.* The supersensor will then send the aggregated image data to the remote Photo tourism center for further processing.

In the example shown in Fig. 4.12, we have chosen three supersensors (denoted in *dark-shaded* circles), named A, B and C. Those un-shaded *red squares* in the upper left part of the field represent the image sensors belonging to the cluster headed by supersensor A. Those un-shaded *black triangles* in the upper right part of the region are the image sensors belonging to the cluster headed by supersensor C. Those un-shaded *blue rhombuses* in the lower part of the region belong to the cluster headed by supersensor B. Notably, each image sensor belongs to *one and only one* cluster.

The problem here is how to find these supersensors in a way such that the overall communication loads among image sensors can be *minimized* while maximizing the number of *common feature points* among images from the same cluster during the image data aggregation process. As we mentioned earlier, the minimization of the overall communication loads also means the minimization of the overall consumed transmission energy among those sensor nodes, each of which is often equipped limited energy resources such as batteries. Intuitively, we need to find sensors that have the most *overlapping view areas* with their nearby neighboring image sensors.

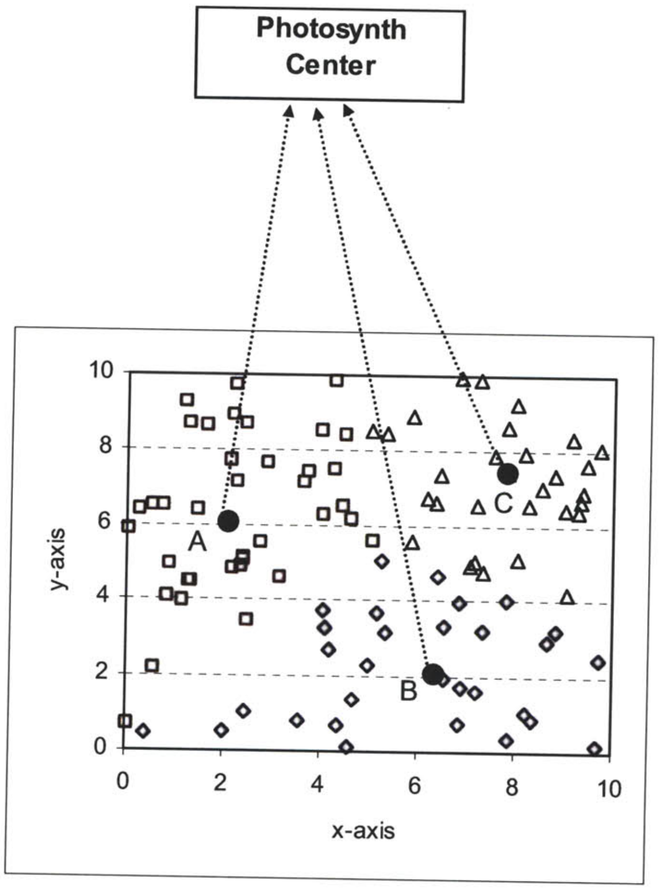


Figure 4.12: An illustration of the image data fusion model.

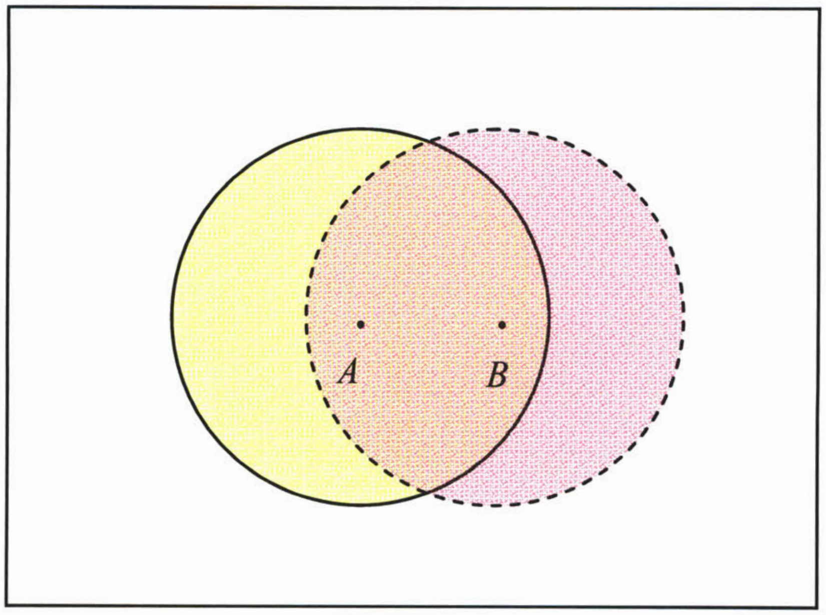


Figure 4.13: The field of view coverage relationship between two neighboring image sensors.

In Fig. 4.13, we show the view coverage relationship between two neighboring image sensors. As we can observe that the closer the two image sensors (sensor *A* and sensor *B*), the more *overlapping* field of view they have. Now let us have a look at a simple example with only three image sensors. It is easy to see that image sensor *B* (the middle one) might be the best candidate to be the supersensor among those three image sensors *only* from the field of view coverage perspective, as the choice of sensor *B* as supersensor potentially leads to the minimal communication loads and the most *common feature points* among images from all three image sensors.

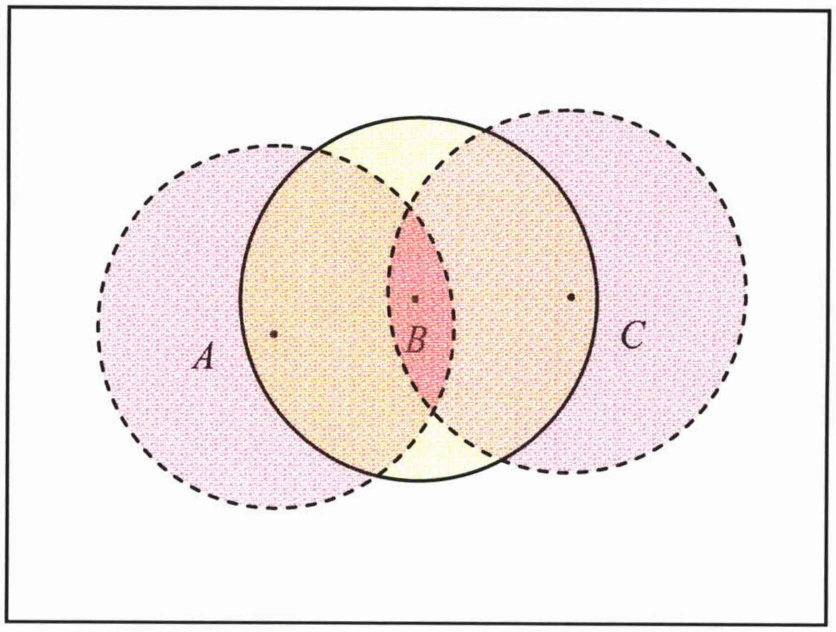


Figure 4.14: An example with three image sensor nodes.

However, in the case of *hundreds* of tiny image sensors it is no longer a trivial problem to choose multiple supersensors based on multiple factors in such a way to minimize the overall communication loads while maximizing the number of *common feature points* among images from the same cluster during the image data aggregation process.

4.2.3.3. The Protocol to Choose Supersensors

The basic idea is to group the image sensor nodes in a given image sensor network into *non-overlapping* clusters such that the "center" (based on the given *cost* function) of each cluster is a chosen supersensor. Each image sensor node belongs to *one and only one* cluster. Assuming that we have a fixed number of clusters, say k of them, at first we can apply the modified version of k-means clustering algorithm [53] with a weighted cost function to find the corresponding k supersensors.

The modified k-means clustering algorithm operates as follows: (a) Randomly initialize k clusters with k centers; (b) Assign image sensor nodes to the clusters with the least cost; (c) Calculate the centroid of each cluster, then move the corresponding center to the new centroid in that cluster; (d) Repeat Step (b) and Step (c) until convergence; (e) Choose the image sensor node that is "closest" to the centroid in each cluster based on the cost function as the corresponding supersensor. After Step *(e)*, we have exactly k supersensors.

4.2.3.4. A Weighted Cost Function

As mentioned earlier, the potential factors that determine the choice of those supersensors include *proximity*, *network distance* (usually measured by the round trip time (RTT)), *communication channel conditions* (usually measured in the form of RSSI (radio signal strength indicator)), and *residual power* at each image sensor, etc. The reason we choose proximity as one of the factors is due to the fact that the closer the two image sensors the more *overlapping* view area they might have in common (see Fig. 4.13).

To accommodate multiple factors, we use a weighted cost function as follows:

$C_{i,j} = \alpha_1 \times d_{i,j} + \alpha_2 \times RTT_{i,j} + \alpha_3 \times \frac{1}{RSSI_j} + \alpha_4 \times \frac{1}{p_j}$, where node j is a supersensor candidate, $d_{i,j}$ indicates the Euclidean distance between node i and node j, $RTT_{i,j}$ stands for the round trip time between node i and node j, $RSSI_j$ is the received signal (from node j) strength indicator by node i, p_j is the residual power at node j, the weights α_is satisfy the equation of $\sum_i \alpha_i = 1$.

4.2.3.5. Distributed Implementation

Assume that we have a total of n image sensor nodes in a given image sensor network and we want to choose k of them as supersensors.

The first step of randomly choosing k centers can be done in a *distributed* way by using a random process that has a fixed probability of $\frac{k}{n}$ in selecting oneself by each image sensor. This means that each sensor node, say node i, picks a uniform random number, say $r[i]$, in the range of $[0,1)$, and then decides that it is an initial supersensor if and only if $r[i] < \frac{k}{n}$. Each supersensor candidate broadcasts a probe packet to its neighboring nodes with the information of the supersensor's name, its residual power, the timestamp in order to calculate *RTT*. Once a non-supersensor candidate receives the probe packet, it will record the *RSSI* and figure out the cost based on the weighted cost function to determine which supersensor it should associate with. After the non-supersensor candidate figures out which supersensor's cluster it belongs to with the least cost, it will send the confirmation to the corresponding supersensor with its own information in order for the supersensor to calculate the new centroid. The adjustment will be made accordingly till its convergence or after a certain number of iterations.

4.2.3.6. Photo tourism Processing

In order to smoothly interpolate images obtained from the image sensors to create Photo tourism experience, we first gather the information of those *feature points* (f_is and

($f_i + \Delta f_i$)s) in each cluster to reconstruct all the images. We then feed those images into Photo tourism to create 3D and 360 degree environment.

4.2.4. Empirical Results

In our simulation, we *randomly* placed 300 image sensor nodes on a 10 by 10 plane. In the following examples, we choose a variety of numbers of supersensors for the same network topology in a variety of circumstances. The empirical results are shown in Fig. 4.15 ~ Fig. 4.22.

In the first set of experiments, we consider the scenarios that all of the image sensors have high residual power while we choose different number of supersensors for the same network topology. In Fig. 4.15 we show the case in which five supersensors are chosen, which are shown in *dark-shaded* squares. The number of image sensors in each of the five clusters is depicted in Fig. 4.16. In Fig. 4.17 we show the case in which ten supersensors are chosen, which are shown in *dark-shaded* squares. The number of image sensors in each of the ten clusters is depicted in Fig. 4.18. From Fig. 4.15 to Fig. 4.18, we observe that with different number of supersensors, the chosen supersensors have changed *accordingly*.

In the second set of experiments, we consider that some of the image sensors have very low residual power while we choose the same number of supersensors for the same network topology. In Fig. 4.19 we show the case in which 20% of the total image sensor nodes have very low residual power (those illustrated in *pink* triangle icons) and those image sensors with very low residual power are chosen *randomly* among all image sensors. The number of image sensors in each of the ten clusters is depicted in Fig. 4.20. In Fig. 4.21 we show the case in which 50% of the total image sensor nodes have very low residual power, which are shown in *pink* triangle icons. The number of image sensors in each of the ten clusters is depicted in Fig. 4.22. From Fig. 4.19 to Fig. 4.22, we observe that with different number of image sensors with very low residual power, the chosen supersensors have changed *accordingly*.

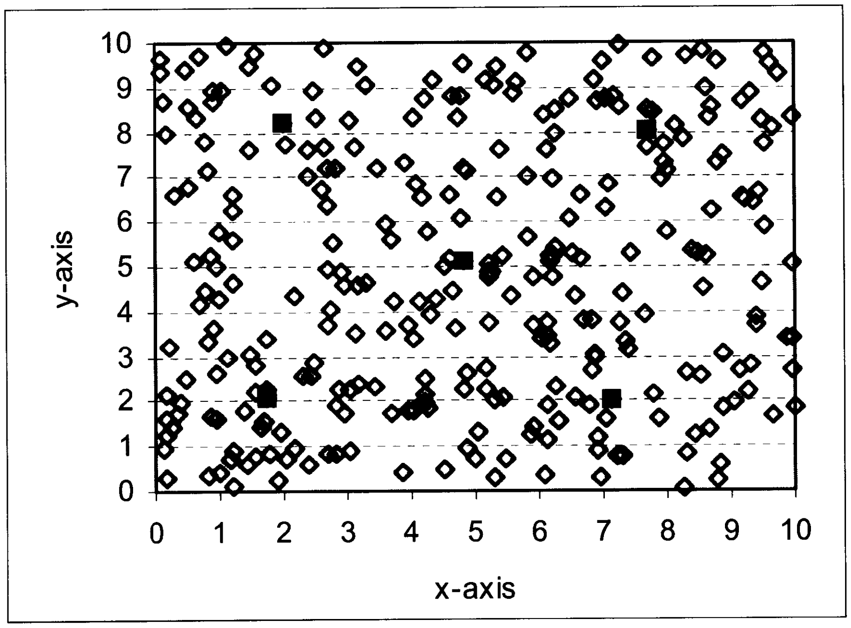


Figure 4.15: The 5 supersensors are shown in dark-shaded squares.

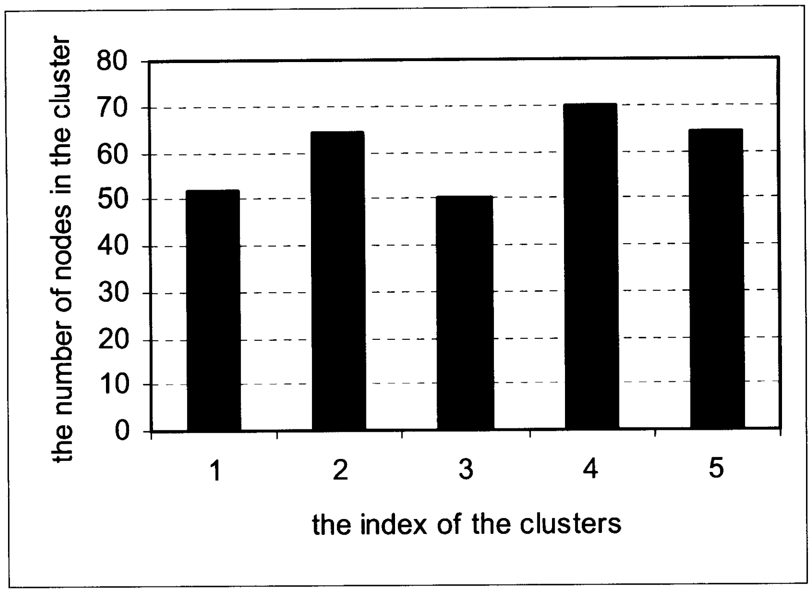


Figure 4.16: The number of image sensors in each cluster.

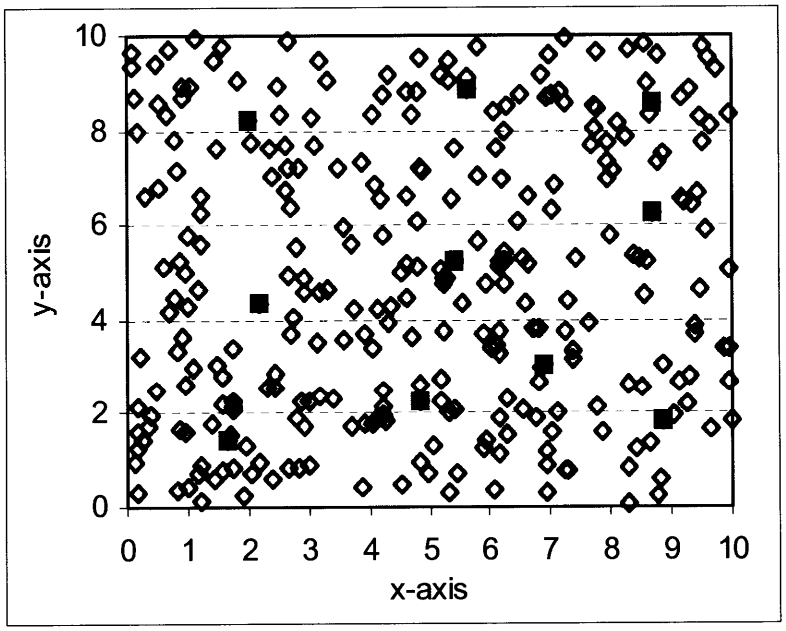


Figure 4.17: The 10 supersensors are shown in dark-shaded squares.

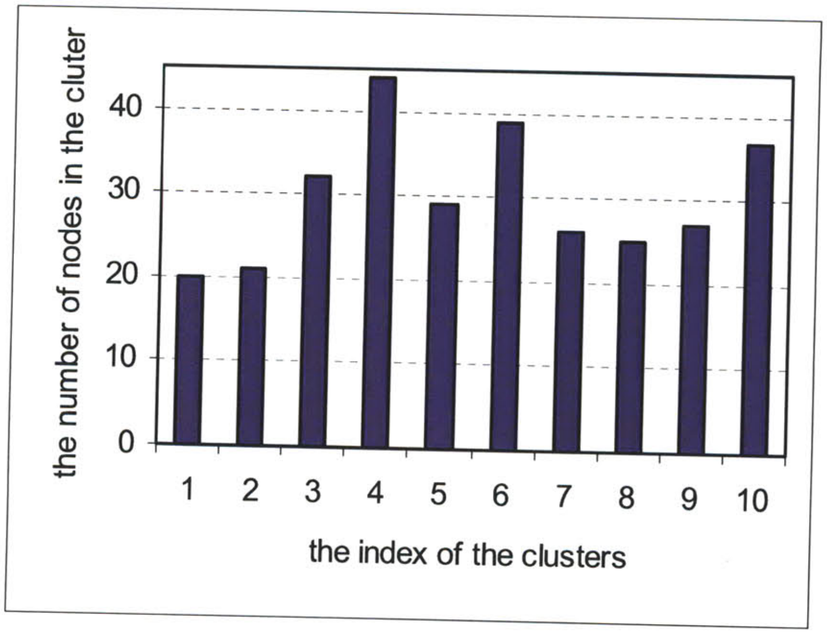


Figure 4.18: The number of image sensors in each cluster.

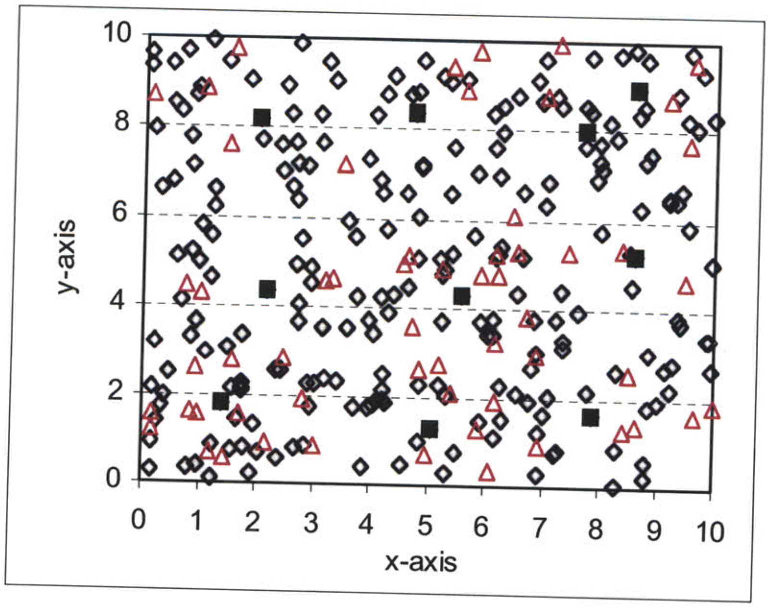


Figure 4.19: The 10 supersensors are shown in dark-shaded squares and those pink triangle icons (20% of them) indicate those with very low residual power.

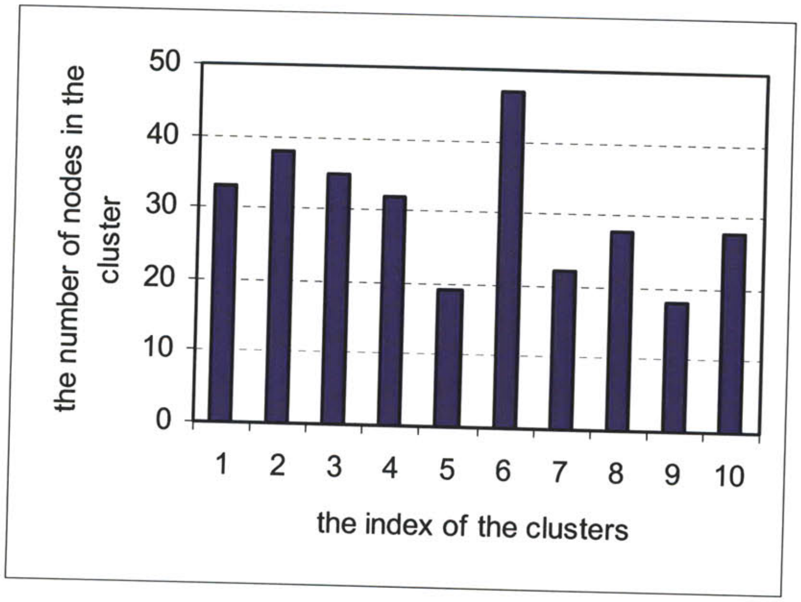


Figure 4.20: The number of image sensors in each cluster.

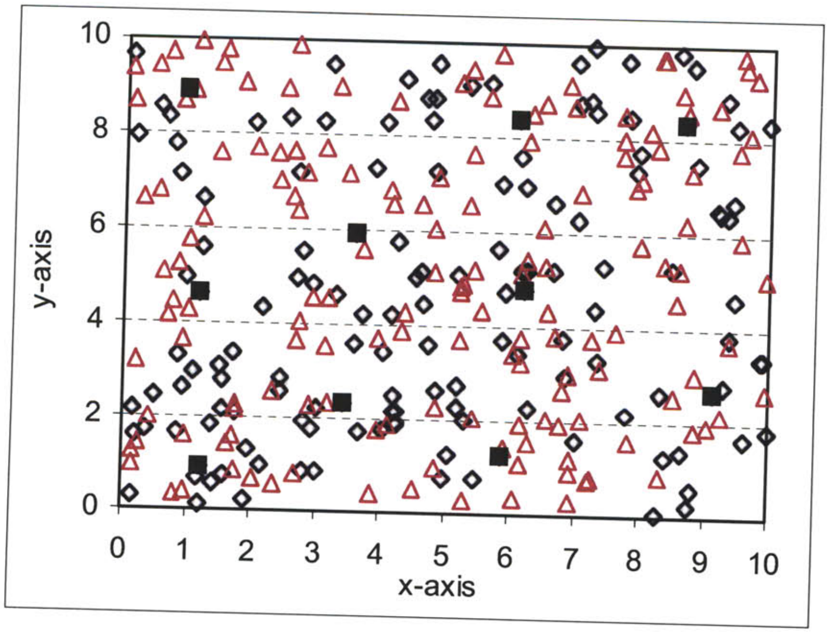


Figure 4.21: The 10 supersensors are shown in dark-shaded squares and those pink triangle icons (50% of them) indicate those with very low residual power.

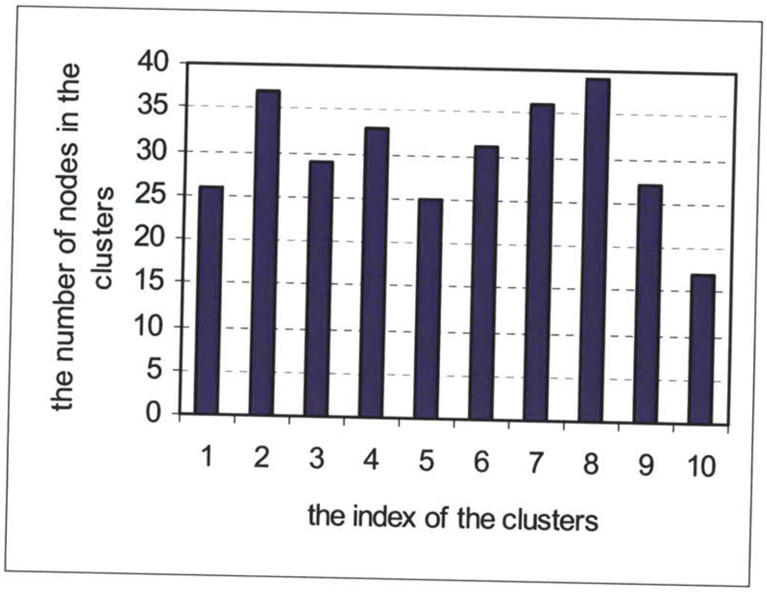


Figure 4.22: The number of image sensors in each cluster.

4.2.5. Summary

In this work we present a novel framework of Photo tourism via wireless image sensing with a group of image sensors. We address several of the key challenges in this new platform in order to achieve even "richer" Photo tourism experience in a simple and efficient way with images generated from image sensors *automatically* and *remotely,* in particular for some *humanly inaccessible* places. In the first step, we remove redundant images whose feature points are already covered by images from neighboring sensors. Then, we choose k supersensors to serve as local image data aggregators to further reduce the redundancy before sending the image data out to the remote Photo tourism center. We presented a distributed algorithm in choosing k supersensors among n image sensors and conducted extensive simulations to examine its adaptability and feasibility. We build image sensor prototypes by attaching our sensor nodes with commodity cameras for a quick demonstration of the concept. Essentially, we propose to conduct clustering and data compression at the sensor network level for the sake of reducing communication cost and energy consumption, instead of doing it at the Photo tourism stage.

Chapter 5

Community Security

In this section, we address the issue of community security in several aspects. First, we present several basic principles in the design of community-based computing systems to safeguard community security. Then, we introduce community-based anti-spam method to deal with spammers as communities instead of dealing with each email spam individually.

5.1. Principles

The first principle is that "don't put all your eggs in one basket". Although this is a well-known quote, people often tend to overlook it in practice. Community storage is a good example to realize this basic principle in community security. In the community storage system, each community member (node) only stores partial encoded information. If you lose your device, it is not a big problem as other people can not see anything meaningful for the stored data on that device. Furthermore, you can retrieve the stored information back from other community members. Lastly, the attacker has to compromise enough nodes in order to reconstruct the original data object.

The second principle is that "treat the bad guys (attackers) as a community". One example is the email spammers. It is reported that email spams already account for more than 97% of the total email traffic [165] and it is hard to imagine that those spammers acted individually. As will be shown in our empirical study that those spammers demonstrated highly organized community behavior. It is far more efficient to block email spams collaboratively among email servers by exchanging the real time community behavior of spammers, instead of dealing each email spam individually. Once the email spams reach the inbox of the email clients, the damage is already done.

5.2: Community-based Anti-Spam Method

In this section, we (joint work with M. Hsieh and P. Gburzynski [90,91]) analyze the community behavior of spammers and propose community-based anti-spam approach to deal with email spams more efficiently.

5.2.1. Spam Data Source

Our spam data was obtained from Jaeyeon Jung and Nick Feamster at CSAIL MIT, where it has been collected at a domain mail server in such a way that the *IP addresses* of the spam sources were recorded when the spammer tried to establish the TCP connection with the domain mail server to deliver the spam message. The IP address of the spammer recorded during the 3-way handshake should be the real IP address of the spammer *in most cases* even though there are rare scenarios in which the spammer could try IP spoofing. In this context, practically the only workable case of IP spoofing involves presenting a formally unused IP address within the subnet on which the spammer's host is actually located. One can argue that in such a case the *de facto* compromised subnet can be viewed as a single source of spam. Another remote alternative is BGP hijacking to propagate fake route entries with unused IP addresses to the nearby ISPs. As BGP hijacking is considerably more difficult (and less predictable) than compromising random hosts in random subnets, we can assume that the impact of such cases on our studies (if present at all) is negligible. Spamming becomes more and more distributed, with the spammer controlling thousands of bots and having several levels of indirection to cover his/ her identity. Consequently, the returns from IP spoofing (which the spammer can seldom bet on) quickly diminish to the point that this technique can be safely ignored. Besides, by blocking email from knowingly unused IP sources, one will never risk a false positive.

The spam mail data contains the full mail header information and the full mail contents including the attachment files. The mail header information contains the *real* IP address of the spam source, the route information (which can be fake up to the real IP address of the actual sender), and the TCP SYN fingerprint, which can be used to identify the OS information of the spam source. The data set covers one week of operation and consists of 86,819 spam messages.

5.2.2. The Community Behavior of Spammers

In this section, we group the spammers in terms of the URLs, stock symbols, monetary amount, which appear in most of the spam messages. Figure 5.1 visualizes the clustering relationships based on the URLs in the spam in day 1. If the same URL appears in a spam message sent from source *A* and source *B* (where *A* and *B* represent IP addresses of two spammers), then an edge is plotted to connect the nodes *A* and *B*. The clustering structure is clear. The number of members in each cluster ranges from 1 to 716.

According to [164], a typical botnet consists of several hundred compromised machines, which is consistent with the sizes of some clusters visible in Figure 5.1. The major component with 716 spammers is further illustrated in Figure 5.2. An interesting observation is that the spam mail at the pivoting point often comes earlier than those homogenous points further away from the cluster's center. Another key observation is that the more groups a spammer's IP address is associated with (due to multiple distinct URLs appeared in the spam mail from this spammer), the higher the probability that spam mail from this IP address will arrive in the near future. We hypothesize that those pivoting points play an important role in the botnet.

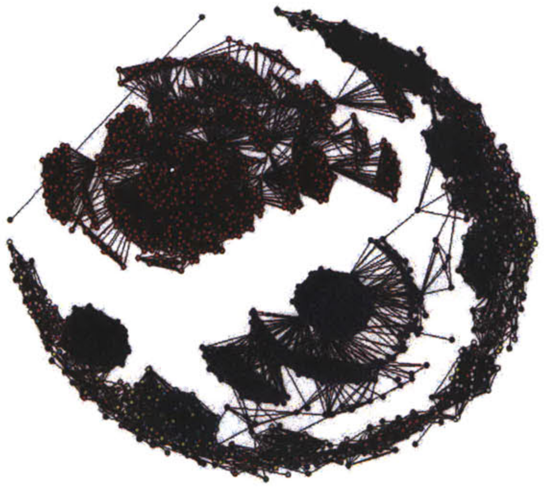


Figure 5.1: The clustering structure of the spammers based on the URLs in spam messages in day 1.

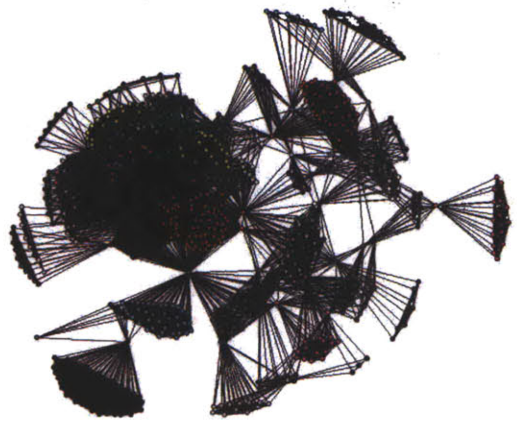


Figure 5.2: The major component of the clustering structure from Figure 5.1.

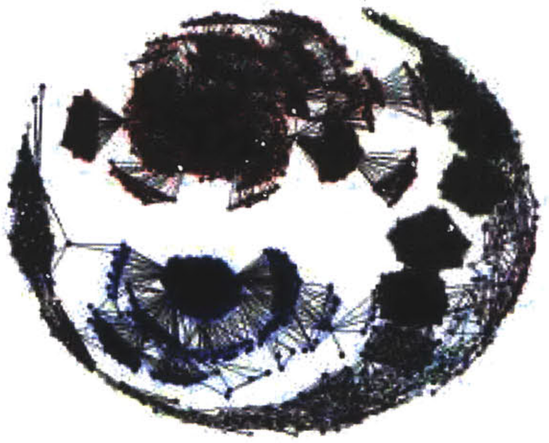


Figure 5.3: The clustering structure of the spammers based on the URLs in spam messages in day 2.

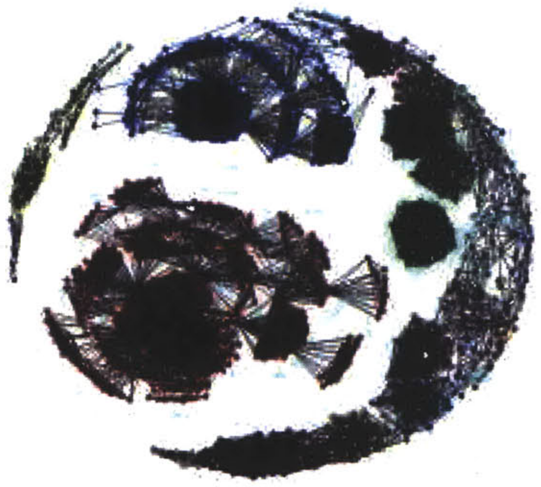


Figure 5.4: The clustering structure of the spammers based on the URLs in spam messages in day 3.

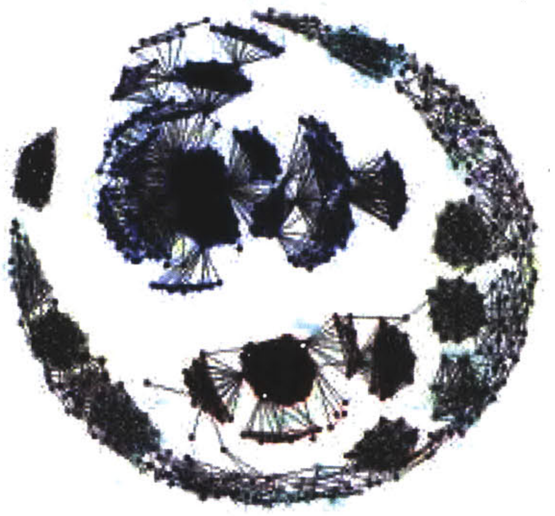


Figure 5.5: The clustering structure of the spammers based on the URLs in spam messages in day 4.

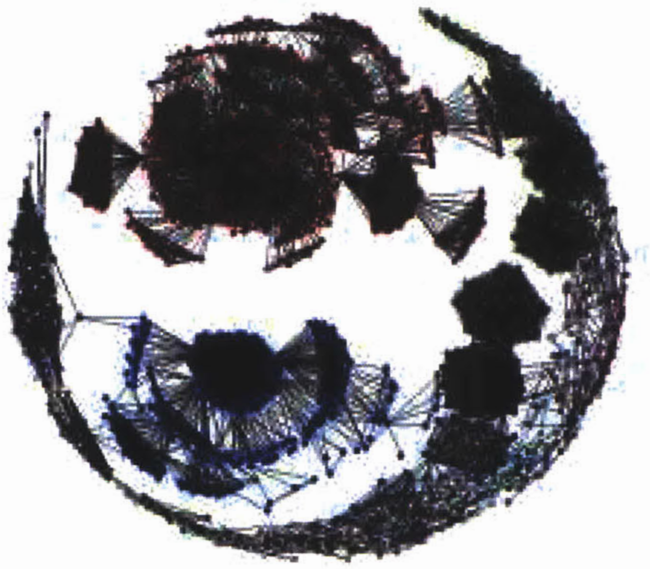


Figure 5.6: The clustering structure of the spammers based on the URLs in spam messages in day 5.

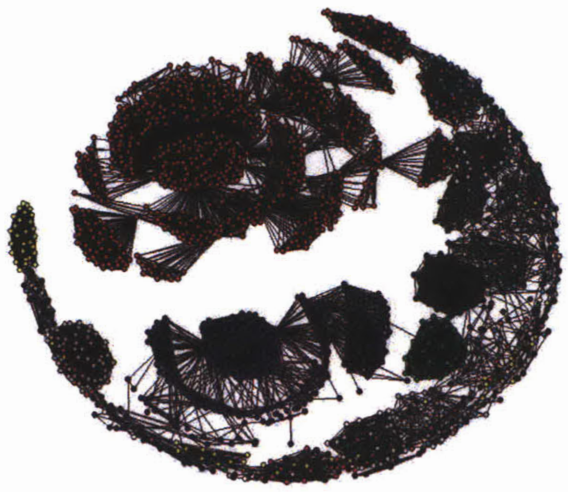


Figure 5.7: The clustering structure of the spammers based on the spam messages arriving in day 6.

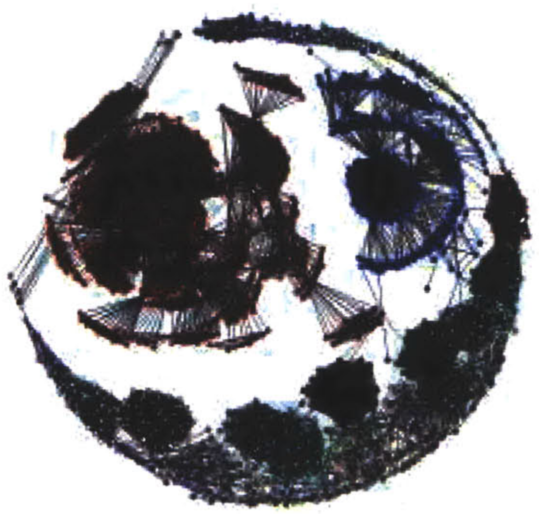


Figure 5.8: The clustering structure of the spammers based on the URLs in spam messages in day 7.

As shown in Fig. 5.3 ~ Fig. 5.8, similar clustering patterns have been observed for the remaining days (Day 2 ~ Day 7).

In Fig. 5.9, we depict the clustering structure of the spammers based on the *money amounts* in spam messages in day 1. As most of the spam is related to money, the clustering structure in Fig. 5.9 seems quite interesting and relevant. Normally, the products, services, commodities, stocks, etc., advertised in spam come with unit prices, so it is unlikely that the spammers would intentionally make those attributes random (e.g., to diversify the signatures of their spam). Consequently, such values are likely to be good discerning and unifying features of spam arriving from different sources. Unfortunately, compared with Figures 5.1 ~ 5.8, we observe that the cluster sizes in this case are relatively small, which hints that this characterization may not work well by itself. Nevertheless, it can be at least applied as a supplementary feature, e.g., together with the URL-based criteria. In Fig. 5.10, we show the clustering structure of the spammers based on *stock symbols* in spam messages in day 1. Clearly, both the number of clusters and the cluster sizes are small. We determine that stock symbol may not be a good criterion to classify the spammers.

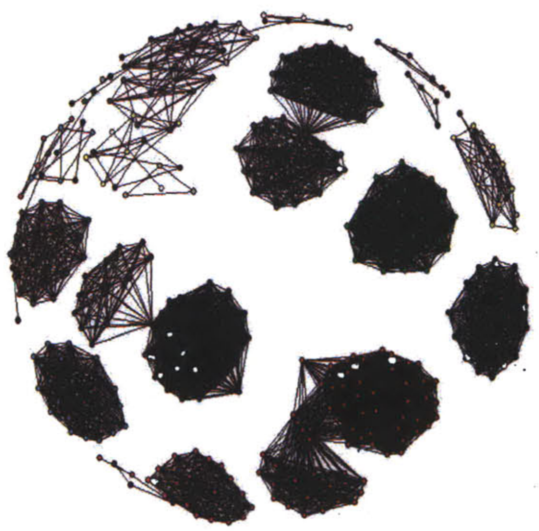


Figure 5.9: The clustering structure of the spammers based on the monetary amounts in day 1.

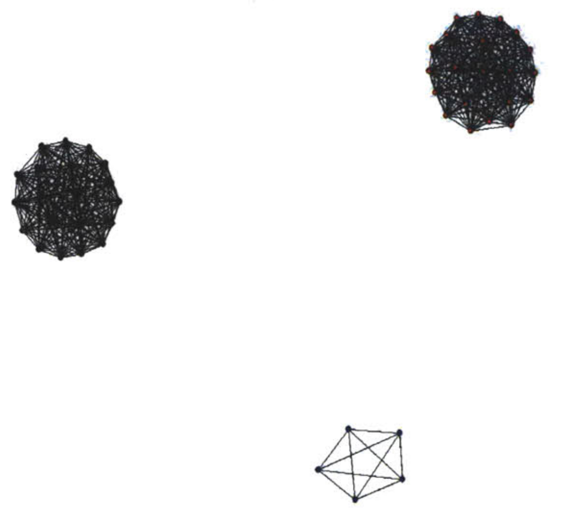


Figure 5.10: The clustering structure of the spammers based on the stock symbols in day 1.

Now, let us examine the correlation coefficient of the inter-arrival time of the spam messages from the spammers belonging to the same cluster. This coefficient is given by following formula [90]:

$$\rho_k = \frac{1}{N-k}\sum_{i=1}^{N-k}\frac{(x_i - \bar{x})(x_{(i+k)} - \bar{x})}{\delta_x^2} \tag{5.1}$$

where N is the number of messages, k is the lag index (the separation between the inter-arrival times being considered), x_i is the i^{th} inter-arrival time, $\bar{x}$ is the average inter-arrival time, δ_x^2 is the variance in the inter-arrival time, and ρ_k is the correlation coefficient of the inter-arrival time with the lag index of k. The intuition is to show that even though some spam arrivals within the same group of spammers are far apart, i.e., their lag index is large, they may still be correlated, which would mean that the messages from the same group of spammers tend to arrive in burst.

As shown in our previous results in [90], spam arrival within the same cluster of spammers exhibits strong long-range dependency and the bursty character of the spam arrival process. The trend is clearly visible despite the somewhat noisy character of the curve. The correlation coefficient oscillates along the "trend line" due to the granularity of the timestamp. This is because, within a given time grain (one second), there are often multiple spam arrivals.

Following [3], given a clustering criterion, we define the clustering coefficient of a spammer as the ratio between the actual number of edges among the neighbors of that spammer node and the number of possible edges among those neighbors. Formally, assume that node i in the network has k_i edges, connecting to k_i other nodes. Let E_i stand for the number of edges that actually exist among these k_i nodes. The total number of possible edges among these k_i nodes would be $k_i(k_i - 1)/2$. The value of the clustering coefficient of node i is thus given by

$$C_i = \frac{E_i}{(k_i \times (k_i - 1))/2} \tag{5.2}$$

The clustering coefficient of the whole network is the average of all the individual C_i's.

The motivation behind the clustering coefficient is to identify the spammers associated with multiple communities (based on the given set of clustering criteria). The idea is to be able to detect the smallest number of spammer responsible for sending the largest amount of spam. Whatever efforts we undertake to identify and block spam, those spammers should receive a "*preferential treatment*."

In Figures 5.11 and 5.12, we show the overall clustering coefficient of the community of spammers for each day in the 7-day period, and the clustering coefficient of the largest component from the community from day 1 to day 7. We also show the total number of spammers in each day and the number of spammers in the largest component for the day. As we can see from Figures 5.11 and 5.12, the overall clustering coefficient for the entire community is very high, indicating the property of small world networks [3]. We also observe that as the community size of the spammers grows, the clustering coefficient decreases slightly. It has been reported that generally collaborative networks such as movie actor networks, co-authorship networks, etc, tend to have higher clustering coefficients [3]. It is thus consistent to say that spammers form a "collaborative" network, organized in the form of BotNet [164], with a high clustering coefficient. This high clustering behavior of spammers bodes well for a community-based approach to spam elimination: it is an argument that dealing with the community as a whole is likely to bring about better results than treating each spammer individually.

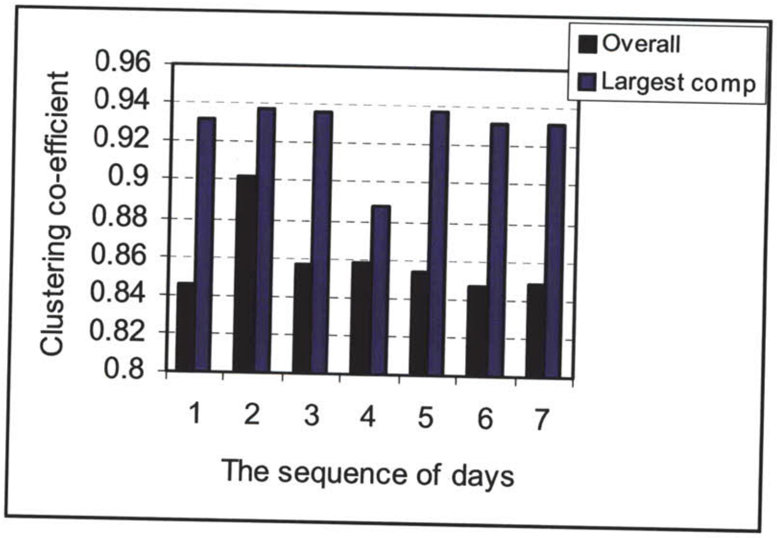


Figure 5.11: The clustering coefficient of the community of spammers from day 1 to day 7.

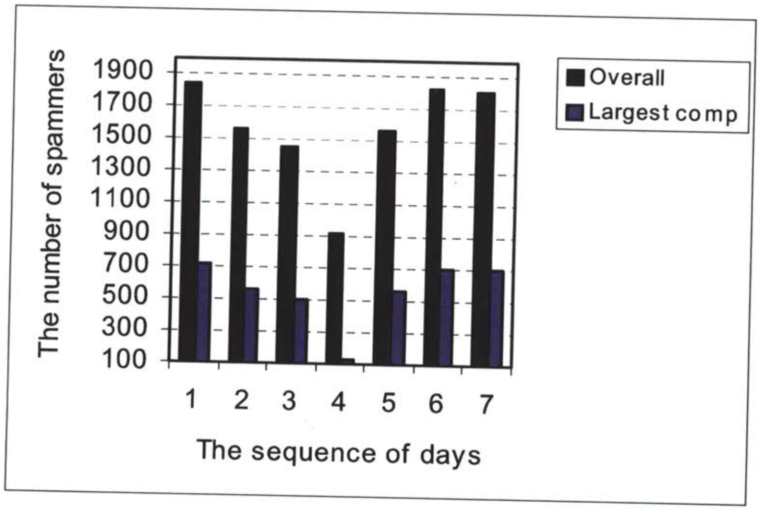


Figure 5.12: The total number of spammers and the number of spammers in the largest component for each day.

5.2.3. Community-based Anti-spam Strategies

In this section we briefly discuss some community-based anti-spam strategies based on the results of our empirical study on the community behavior of spammers in Section 5.2.2. The community-based anti-spam framework can be used as a *complementary* component for an existing anti-spam system, e.g., SpamAssasin, to efficiently block spams from organized spammers. The idea is that if we can perceive some community-based behavior/patterns of spam senders based upon some common signatures from the email content and/or headers, e.g., URL or some other criteria, we can assign a high spam score to the messages from this group. The more members in the given suspected spammer group, the higher spam scores for the messages from that group. The intuition behind this is that it is *highly unlikely* for a large group of legitimate senders to send emails with exactly the same type of signatures, e.g., the same URL. The mail servers can also act as a community to exchange real time community behavior of spammers to block email spams more efficiently.

The source IP address of an incoming message is used as a unique identifier of the email sender for community-based classification. We do not use the IP address of the email sender for blacklist blocking. So, even for rare cases of spoofed source IP address, it will have little impact on the effectiveness of our group-based anti-spam approach.

When an email comes, we first extract all the potential advertisement/commercial URLs from the mail content, then calculate the hash value for each URL in the given email and update the number of members counter based on distinct IP addresses for the corresponding associated groups. Finally, we update the number of associated groups counter for the given IP address of the mail sender. We use a time sliding window exponentially weighted moving average to calculate the average number of members for a given group. The detailed algorithm is described in Section 5.2.4. We say a new group is terminated if and only if the average number of members for a given group is below a given threshold, say $G_{\exp}$. Once a group is terminated, the original group has to be dismantled and the original members of this group have to update their state accordingly. Based on the number of members of each group, we assign a spam score for the given email. In this preliminary study, we assign a blocking probability for the given email based on the number of members in that group and the number of associated groups for a given spammer.

5.2.4. TSW-EWMA Algorithm

We use a time sliding window exponentially weighted moving average (TSW-EWMA) algorithm [167] to *dynamically* calculate the state of each URL-based group and determine if an old group should be terminated based on the spam arrivals. Let W be the window size in terms of some time unit, say hours. Let $I_{URLi-pre}$ and $I_{URLi-cur}$ denote the number of members, e.g., the number of distinct IP address, associated with this URL-based group, in the previous time window and the current time window respectively. Let I_{URLi} be the average number of members for the given URL. We have

$$I_{URLi} = \alpha \times I_{URLi-pre} + (1-\alpha) \times I_{URLi-cur} \tag{5.3}$$

where α is the weight, e.g., the filter constant. This type of moving average filter places more importance to more recent email data by discounting older mail data in an exponential manner. At the same time, the filter smoothes transient behavior of email traffic.

5.2.5. A Simple Hash Lookup

First, we convert the ASCII characters of a given URL into binary data format and let x_i denote the number represented by the bits in the i^{th} character of the URL. Notably, each ASCII character is represented with a distinct 7-bit binary data. Let m be a large number, say, $m > 2^7 \times L_{\max}$, where $L_{\max}$ is the maximum length with respect to the number of characters for a given URL, say, 80. We define a simple hash function as follows:

$$H(URL) = \sum_{i=1}^{n} x_i \bmod \mathrm{m}, \tag{5.4}$$

where n is the number of characters in the given URL. We use chaining, e.g., a link bucket, for hash function collision resolution.

Alternatively, we can use a SHA-1-based hash function, which takes an arbitrary length of URL (less than 2^{64} bits in length) as input and produces a 160-bit number as the corresponding URL digest. The SHA-1-based hash function has the well-known property of collision resistance. The drawback is that SHA-1-based hash function is more *computationally* expensive than the one described in Eq. (5.4).

5.2.6. Preliminary Results

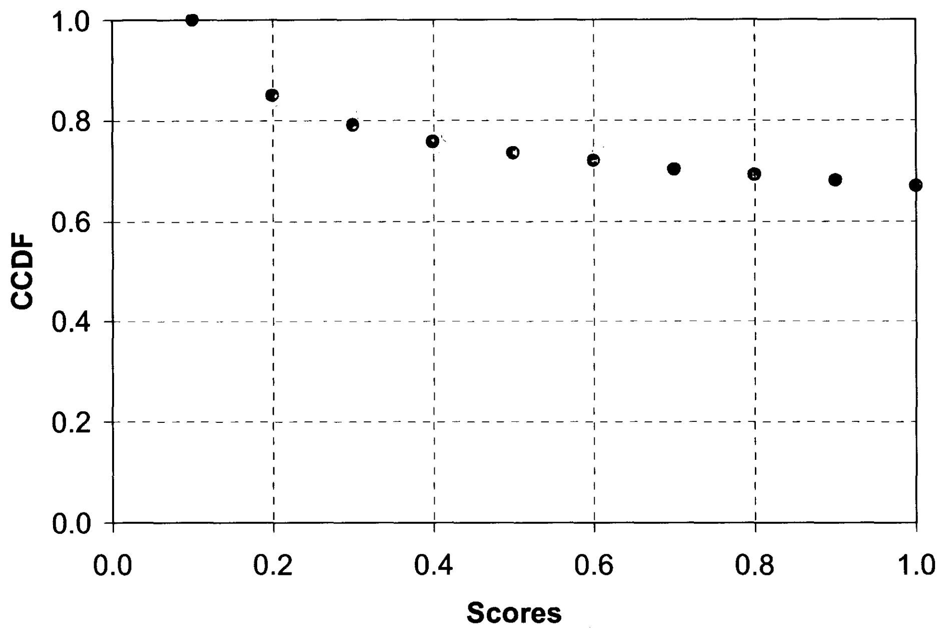


Figure 5.13: The spam score distribution based on community-based approach.

As shown in Fig. 5.13, the x-axis indicates the spam scores and the y-axis denotes the CCDF (complementary cumulative probability) of the spam score. Clearly, our preliminary results show that the group-based approach can block 70% to 90% of the spams, depending on the implementation parameters. To the best of our knowledge, it is the first time that community-based anti-spam method has been explored.

5.2.7. Issues and Open Challenges

There are several challenges for group-based anti-spam strategies based on URL-grouping. For example, if a host is running DHCP, the host IP address could change from several hours to several days. The change of the IP addresses could change the clustering structure of the spammers.

Another issue is that the spammers could use various HTTP formats to intentionally hide the URL information. For example, the following links all indicate the same link as http://www.yahoo.com: http://3631052355 (a single decimal number of the IP address), http://0xD86D7643 (a single hexadecimal number of the IP address) http://0330.0155.0166.0103 (the dotted form in octal) [166].

The spammers can also use some steganography techniques such as html color, graph, etc. to camouflage URL or other information used to group the spammers.

5.2.8. Summary

In this work we have investigate the community structures of spammers based on collection of spam traffic sighted at a domain mail server. Our study shows that the relationship among spammers demonstrates highly clustering structures based on URL-grouping. The inter-arrival time of spams from the same group of spammers exhibits long-range dependency in the sense that the spam messages from the same group of spammers often arrive in burst. We also observe that spammers associated with multiple groups tend to send more spam messages in the near future. Finally, the high clustering coefficient of the community of spammers reveals the "collaborative" nature of those spammers.

Finally, we would like to thank Jaeyeon Jung and Nick Feamster for the spam data sources used in this study and we also thank Jaeyeon Jung for her invaluable input and suggestion.

Chapter 6

Contributions

In this thesis, we lay foundations for a distributed community-computing environment. We have made significant contributions in several fronts: community coding, community storage, community sensing, and community security.

We introduce community coding to model a wide range of applications in the new paradigm of community computation. The community coding approach should be applicable to many challenging problems in this new paradigm, where information systems meet social networks.

We propose a network-aware source coding method with multiple sources in the framework of community coding. We also present community coding formulations for P2P media streaming with application level multicast to maximize the sum of the utilities of all the peer nodes, given their heterogeneous resource constraints. We present a novel erasure code called community codes, which is rateless, locally encodable and locally decodable, for a wide range of applications such as reliable data communication, distributed data storage, etc.

We build the prototype of a shared e-Library for a large, secure and distributed community storage system among a group of community member nodes. We also build the prototype of a collaborative offline Wikipedia among a group of community member nodes, where the Internet connection may not available. We conduct extensive experiments to quantitatively and qualitatively measure the performance of the community storage system with two main metrics: the availability rate and the uploading/retrieving delay.

We have made significant contributions in the area of community sensing in two areas: view coverage with multiple image sensors, and an efficient, distributed method for image data aggregation with a group of image sensors. We analyze view coverage with random dropping of image sensors in a variety of scenarios and the analysis provides view coverage assurance with a certain probability, which is the key for the performance of such image sensor networks. We present a distributed clustering algorithm for efficient image data aggregation among a group of image sensors and we conduct extensive empirical studies for the performance of the proposed approach.

We present several basic principles for community security. As one application example, we empirically analyze the community behavior of spammers. We propose community-based anti-spam approaches to deal with email spams much more efficiently than existing anti-spam tools.

Chapter 7

Conclusions

In this thesis we lay the foundations for a distributed, community-based computing environment to tap the resources of a community to better perform some tasks, either computationally hard or economically prohibitive, or physically inconvenient, that one individual is unable to accomplish efficiently. We introduce community coding, where information systems meet social networks, to tackle some of the challenges in this new paradigm of community computation.

The central thread of this thesis is the community coding optimization framework that we introduced. Community coding is an emerging field of social computation and coding theory, which provides a method of aggregating resources from a community to maximize the sum of their utilities by allowing the community members to coordinate their contributions of resources based on some predefine rules such as proportional fairness and minimum utility rules.

We design algorithms, protocols and build system prototypes to demonstrate the power of community computation to better deal with reliability, scalability and security issues, which are the main challenges in many emerging community-computing environments, in several application scenarios such as community storage, community sensing and community security. For example, we develop a community storage system that is based upon a distributed P2P (peer-to-peer) storage paradigm, where we take an array of small, periodically accessible, individual computers/peer nodes and create a secure, reliable and large distributed storage system. The goal is for each one of them to act as if they have immediate access to a pool of information that is larger than they could hold themselves, and into which they can contribute new stuff in a both open and secure manner. Such a contributory and self-scaling community storage system is particularly useful where reliable infrastructure is not readily available in that such a system facilitates easy ad-hoc construction and easy portability. In another application scenario, we develop a novel framework of community sensing with a group of image sensors. The goal is to present a set of novel tools in which software, rather than humans, examines the collection of images sensed by a group of image sensors to determine what is happening in the field of view. We also present several design principles in the aspects of community security. In one application example, we present community-based email spam detection approach to deal with email spams more efficiently.

In our daily lives, communities like to share things. People share photos on Flikr [169], share videos on YouTube [170], and even share bandwidth via VidTorrent [171]. People also like to help each other, reward each other and look after each other's welfare. In this thesis, we present a set of tools to help to tap the resources of a community for the benefits of all the community members. We hope that the theory of community coding that we introduced and the suite of tools that we presented will help to pave the way for a new way of thinking about clouds and shared resources among a community.

Bibliography

[1] Abbasfar, A., D. Divsalar, K. Yao, “Accumulate-Repeat-Accumulate Codes”, in IEEE Trans. on Comm., April 2007.

[2] Aguilera, M., R. Janakiraman, L. Xu, “Using erasure codes for storage in a distributed system”, in the proceeding of *DSN* ‘2005.

[3] Albert, R., A. Barabasi, “Statistical Mechanics of Complex Networks”, *Reviews of Modern Physics,* 74, 47-97 (2002).

[4] Aliaga, D., I. Carlbom, "Plenoptic Stitching: A Scalable Method for Reconstructing 3D Interactive Walkthroughs", *Proceedings of ACM SIGGRAPH*, Aug., 2001.

[5] Aliaga, D., D. Yanovsky, T. Funkhouser, I. Carlbom, "Interactive Image-Based Rendering Using Feature Globalization", *Proceedings of ACM Symposium on Interactive 3D Graphics*, Apr., 2003.

[6] Anderson, R., “The Eternity Service”, in the proceeding of *Pragocrypt ‘1996.*

[7] Badishi G., I. Keidar, R. Melamed, “Towards Survivability of Application-Level Multicast”, in *FuDiCo II*, June 2004.

[8] Bairavasundaram, L., G. Goodson, S. Pasupathy, J. Schindler, “An Analysis of Latent Sector Errors in Disk Drives”, in the proceeding of *ACM Sigmetrics ‘2007.*

[9] W. Bajwa, J. Haupt, A. Sayeed, R. Nowak, “Compressive Wireless Sensing”, ACM *IPSN ‘2006.*

[10] Balister P., A. Sarkar, B. Bollobas, S. Kumar, “Reliable Density Estimates for Achieving Coverage and Connectivity in Thin Strips of Finite Length”, in *ACM Mobicom* ‘2007.

[11] Banerjee, S., B. Bhattacharjee and C. Kommareddy, "Scalable Application Layer Multicast", in *SIGCOMM '2002.*

[12] Barabas, J., "Sensor Planning for Novel View Generation by Camera Networks", Master Thesis, the Media Laboratory, MIT, 2006.

[13] R. Baraniuk, "Compressive Sensing", in *IEEE Signal Processing Magazine*, July 2007.

[14] Bar-Ilan, J., D. Zernik, "Random Leaders and Random Spanning Trees", in Proc. of the 3rd International Workshop on Distributed Algorithms, 1989.

[15] Barr K., K. Asanovic, "Energy-Aware Lossless Data Compression", in the Proc. of ACM MobiSys 2003.

[16] Baset S., H. Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol", in IEEE *INFOCOM '2006.*

[17] Bergmans, P., "Random Coding Theorem for Broadcast Channels with Degraded Components", in IEEE Transaction on Information Theory, March 1973.

[18] Bertsimas, D., J. Tsitsiklis, "Introduction to Linear Optimization", *Athena Scientific Press*, 1997.

[19] Bhagwan, R., K. Tati, Y. Cheng, S. Savage, G. Voelker, "Total recall: System support for automated availability management", in the proceeding of *NSDI*, 2004.

[20] BitTorrent: http://www.bittorrent.com/

[21] Blake, C., R. Rodrigues, "High availability, scalable storage, dynamic peer networks: Pick two", in the proceeding of *HOTOS*, 2003.

[22] Bolosky, W., J. Douceur, D. Ely, M. Theimer, "Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs", in the proceeding of ACM Sigmetrics, 2000.

[23] Buragohain, C., D. Agrawal and S. Suri, "A Game-Theoretic Framework for Incentives in P2P Systems", in Proc. of IEEE P2P '2003.

[24] Burrows, M., D. Wheeler, "A block sorting lossless data compression algorithm", Technical Report 124, Digital Equipment Corporation, 1994.

[25] Byers J., J. Considine, M. Mitzenmacher, S. Rost, "Informed Content Delivery Across Adaptive Overlay Networks", in IEEE *Trans. on Networking*, Oct. 2004.

[26] Byers, J., M. Luby, M. Mitzenmacher, "Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads", in the proceeding of *IEEE Infocom '1999.*

[27] Camp, T., J. Boleng, V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research", in journal of Wireless Communication and Mobile Computing, special issue on mobile ad hoc networking, vol. 2, no. 5, 2002.

[28] Castro, M., et al, "Splitstream: High-bandwidth Multicast in Cooperative Environments", in Proc. ACM SOSP '2003.

[29] Cataltepe, Z., P. Moghe, "Characterizing Nature and Location of Congestion on the Public Internet", in Proc. ISCC '2003.

[30] Chakrabarti, D., J. Leskovec, C. Faloutsos, S. Madden, C. Guestrin, M. Faloutsos, "Information Survival Threshold in Sensor and P2P Networks", in the proceeding of *IEEE Infocom '2007.*

[31] Chou, P., V. N. Padmanabhan, H. J. Wang, "Resilient peer--to--peer streaming", Technical Report MSR-TR-2003-11, Microsoft Research, 2003.

[32] Chu, Y., S. G. Rao, S. Seshan, H. Zhang, "A Case for End System Multicast", in IEEE JSAC, vol 20(8), 2002.

[33] Chun, B., F. Dabek, A. Haeberlen, E. Sit, H. Weatherspoon, F. Kaashoek, J. Kubiatowicz, R. Morris, "Efficient replica maintenance for distributed storage systems", in the proceeding of *NSDI*, 2006.

[34] Cipar, J., M. Corner, E. Berger, "TFS: A Transparent File System for Contributory Storage", in the proceeding of *USENIX FAST '2007.*

[35] Cohen, B., "Incentives build robustness in BitTorrent", in Proc. 1st Workshop on the Economics of Peer-to-Peer Systems, 2003.

[36] Criado, R., et al, "Choosing a leader on a complex network", in Journal of Computational and Applied Mathematics, July 2007.

[37] Csiszar I., J. Korner, "Information Theory: Coding Theorems for Discrete Memoryless Channels", In Probability and Mathematics Statistics, New York, Academic, 1981.

[38] Cuff, P., "Communication Requirements for Generating Correlated Random Variables", in Proc. of IEEE ISIT '2008.

[39] Dabek, F., F. Kaashoek, D. Karger, R. Morris, I. Stoica, "Wide-area cooperative storage with CFS", in the proceeding of ACM SOSP '2001.

[40] Datta, A., K. Aberer, "Internet-scale Storage Systems under Churn – A Study of the Steady-State using Markov Models", in the proceeding of *IEEE P2P '2006.*

[41] Decuetos P. and K.W. Ross, “Unified Framework for Optimal Video Streaming”, in IEEE *INFOCOM ‘2004.*

[42] Deshpande, H., M. Bawa, H. Garcia-Molina, “Streaming Live Media over a Peer-to-Peer Network”, Technical Report, Stanford University, 2001.

[43] Dimakis , A., P. Godfrey, M. Wainwright and K. Ramchandran "Network Coding for Distributed Storage Systems” , in the proceeding of *IEEE Infocom 2007.*

[44] Dimakis, A., V. Prabhakaran, K. Ramchandran , “Decentralized Erasure Codes for Distributed Networked Storage”, in the proceeding of *ACM/IEEE IPSN ‘2005.*

[45] Dimakis, A., V. Prabhakaran, K. Ramchandran, “Distributed Fountain Codes for Networked Storage”, in *IEEE Transactions on Information Theory*, June 2006.

[46] Dimakis, A., A. D. Sarwate, M. J. Wainwright, “Geographic Gossip: Efficient Aggregation for Sensor Networks”, in Proc. of IPSN ‘2006.

[47] Diot, C., et al, “Deployment Issues for the IP Multicast Service and Architecture”, IEEE Network, vol. 14(1), 2000.

[48] M. Duarte, S. Sarvotham, D. Baron, M. Wakin, R. Baraniuk, “Performance Limits for Jointly Sparse Signals via Graphical Models”, Proc. of *SenSIP* ‘2008.

[49] Estrin, D., R. Govindan, J. Heidemann, S. Kumar, “Next Century Challenges: Scalable Coordination in Sensor Networks”, in Proc. of ACM Mobicom ’99.

[50] Etesami, O., A. Shokrollahi, “Raptor Codes on Binary Memoryless Symmetric Channels”, in IEEE Trans. on Info. Theory, May 2006.

[51] Feldman, M., C. Papadimitriou, J. Chuang, I. Stoica, “Free-Riding and Whitewashing in Peer-to-Peer Systems”, in Proc. SIGCOMM PINS, 2004.

[52] Fowler R., M. Paterson and S. Tanimoto, "Optimal Packing and Covering in the Plane Are NP Complete", in *Information Processing Letters*, Vol. 12, No. 3, 1981.

[53] J. Galagan, "Clustering", *Lecture notes*, MIT class 6.878, Fall 2007.

[54] Gallager, R., "A Simple Derivation of the Coding Theorem and Some Applications", in IEEE Trans. on Info. Theory, 1964.

[55] Gallager, R., "Coding for Discrete Sources", Lecture notes of Principle of Digital Communications, MIT.

[56] Gao Y., K. Wu, F. Li, "Analysis on the Redundancy of Wireless Sensor Networks", in *ACM WSNA* '2003.

[57] Ganesh, A., A.-M. Kermarrec, L. Massoulie, "Peer-to-Peer Membership Management for Gossip-Based Protocols", in IEEE Tran. Comp., vol. 52(2), 2003.

[58] Ghodsi, A., S. El-Ansary, S. Krishnamurthy, and S. Haridi, "A Self-stabilizing Network Size Estimation Gossip Algorithm for Peer-to-Peer Systems", *SICS Technical Report T2005:16. Swedish Institute of Computer Science ISSN 1100-3154.*

[59] Glover F., "Future Paths for Integer Programming and Links to Artificial Intelligence", in *Computers and Operations Research*, 1986.

[60] Godfrey, D., S. Shenker, I. Stoica, "Minimizing churn in distributed systems", in the proceeding of *ACM SIGCOMM '2006.*

[61] Goldsmith A.J., S. B. Wicker, "Design Challenges for Energy-Constrained Ad Hoc Wireless Networks", IEEE Wireless Communication, Aug. 2002.

[62] Goyal, V., "Multiple Description Coding: Compression Meets the Network", in IEEE Signal Processing Magazine, 2001.

[63] Grosu, D., A. Chronopoulos, M. Leung, "Load Balancing in Distributed Systems: An Approach Using Cooperative Games", in the proceeding of *IEEE IPDPS '2002.*

[64] Gummadi, K., S. Saroiu, S. Gribble, "King: Estimating Latency between Arbitrary Internet End Hosts", in Proc. ACM SIGCOMM IMW, 2002.

[65] Gupta, M., P. Judge, M. Ammar, "A Reputation System for Peer-to-Peer Networks", in Proc. of ACM NOSSDAV, 2003.

[66] Gupta P., P. R. Kumar, "The Capacity of Wireless Networks", in IEEE Trans. on Information Theory, March, 2000.

[67] Guruswami, V., P. Indyk, "Expander-based Construction of Efficiently Decodable Codes", in the Proc. of IEEE FOCS '2001.

[68] Habib, A., J. Chuang, "Incentive Mechanism for Peer-to-Peer Media Streaming", in Proc. of IEEE IWQoS, 2004.

[69] Haeberlen, A., Mislove, A., Druschel, P., "Glacier: Highly durable, decentralized storage despite massive correlated failures", in the Proc. of NSDI '2005.

[70] Haiman, M., "Notes on Reed-Solomon Codes", http://math.berkeley.edu/~mhaiman */math55/reed-solomon.pdf*

[71] Hart, M., "Project Gutenberg", http://www.gutenberg.org

[72] Heinzelman, W., A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks", in Proc. of IEEE HICSS '2000.

[73] Ho, T., "Summary of Raptor Codes", Oct. 2003.

[74] Hoory, S., N. Linial, A. Wigderson, "Expander Graphs and Their Applications", in Bulletin of American Mathematical Society, Aug. 2006.

[75] Huang, C., M. Chen, J. Li, "Pyramid Codes: Flexible Schemes to Trade Space for Access Efficiency in Reliable Data Storage System", in the proceeding of *IEEE NCA* '2007.

[76] Huffman D. A., "A Method for the Construction of Minimum-Redundancy Codes", in proceeding of IRE, 1952.

[77] Jennings C., D. Bryan, "P2P For Communications: Beyond File Sharing", in Business Communication Review, Feb. 2006.

[78] Kamvar, S., M. Schlosser, H. Garcia-Molina, "The Eigentrust Algorithm for Reputation Management in P2P Networks", in Proc. of ACM WWW, 2003.

[79] Kim, J., M. Cetin, A. Willsky, "Nonparametric Shape Priors for Active Contour-based Image Segmentation", in Elsevier Journal of Signal Processing, June 2007.

[80] Koizumi, S., H. Ishiguro, "Town Digitizing: Omnidirectional Image-Based Virtual Space", in LNCS, 2005.

[81] Kostic, D., A. Rodriguez, J. Albrecht, A. Vahdat, "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh", in Proc. ACM SOSP, 2003.

[82] Kubiatowicz, J., et al, "Oceanstore: An Architecture for Global-Scale Persistent Storage", in the proceeding of *ASPLOS* '2000.

[83] Larsen R., M. Marx, "An Introduction to Mathematical Statistics and Its Application", Prentice-Hall, 2000.

[84] Levin A., W. Freeman, F. Durand, "Understanding Camera Trade-offs through a Bayesian Analysis of Light Field Projections", in *IEEE ECCV* '2008.

[85] Levis, P., N. Patel, D. Culler, S. Shenker, "Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks", in Proc. of NSDI '2004.

[86] Li, F., R. Raskar, A. Lippman, "Analysis on Probabilistic View Coverage for Image Sensing -- A Geometric Approach", in the *Proceedings of* IEEE IPCCC, Phoenix, Arizona, USA, 2008.

[87] Li, F., D. P. Reed, A. Lippman, "Collaborative Storage with Mobile Devices in Wireless Networks for P2P Media Sharing", *in the proceeding* IEEE WTS, Pomona, California, USA, 2008.

[88] Li, F., J. Huang, A. Lippman, "A Linear Integer Programming Approach to Analyze P2P Media Streaming", *in the proceeding of IEEE* CISS, Princeton University, NJ, USA, March 2008.

[89] Li, F., "The Expansion Rate of Margulis Expanders and LPS Expanders for Vertex Set ZxZ", *in the proceeding of* CISS, Princeton University, NJ, USA, March 2008.

[90] Li, F., M. Hsieh, "An Empirical Study of Clustering Behavior of Spammers and Group-based Anti-Spam Strategies", in the *Proceedings of the 3rd International Conference on Email and Anti-Spam* (CEAS), California, USA, July 2006.

[91] Li, F., M. Hsieh, P. Gburzynski "Understanding the Community Behavior of Spammers", Technical Report, the Media Laboratory, MIT, 2008.

[92] Li, F., "NASC: Network-Aware Source Coding for Wireless Broadcast Channels with Multiple Sources", in the *Proceedings of* IEEE VTC, Montreal, Canada, Sept. 2006.

[93] Li, F., J. Barabas, A. Santos, R. Gens, "Live Photo Mosaic via Wireless Image Sensing with A Group of Image Sensors", Technical Report, MIT Media Lab, 2009.

[94] Li, F., J. Barabas, R. Raskar, A. Mohan, "Reducing Error Due to Photon Noise and Quantization with Multiple Images", Technical Report, MIT Media Lab, 2009.

[95] Li J., C. Blake, D. S. J. De Couto, H. I. Lee, R. Morris, "Capacity of Ad Hoc Wireless Networks", in the *proc. of ACM Mobicom '2001.*

[96] Liu B., P. Brass, O. Dousse, "Mobility Improves Coverage of Sensor Networks", in *ACM Mobihoc* '2005.

[97] Lo, V., D. Zhou, Y. Liu, C. Dickey, and J. Li, "Scalable Supernode Selection in Peer-to-Peer Overlay Networks", in Proc. of IEEE HOT-P2P '2005.

[98] Luby, M., "LT Codes", in the proceeding of *IEEE FOCS* '2002.

[99] Luby, M., M. Mitzenmacher, A. Shokrollahi, D. Spielman, "Improved Low-Density Parity-Check Codes Using Irregular Graphs", in IEEE Trans. on Info. Theory, Feb. 2001.

[100] Magharei, N., R. Rejaie, "Understanding Mesh-based Peer-to-Peer Streaming", in Proc. of ACM NOSSDAV, 2006.

[101] Magharei, N., R. Rejaie, Y. Guo, "Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches", in Proc. of IEEE INFOCOM, 2007.

[102] Magharei, N., R. Rejaie, "PRIME: Peer-to-Peer Receiver-drIven Mesh-based Streaming", in Proc. of IEEE INFOCOM, 2007.

[103] Mallett, J., V. M. Bove Jr., "Eye Society", *in the Proc. of IEEE ICME '2003.*

[104] Maymounkov, P., D. Mazieres, "Rateless Codes and Big Downloads", in the Proc. of the 2nd International Workshop on P2P Systems, 2003.

[105] Maymounkov, P., "Online Codes", Technical Report, New York University, Nov. 2002.

[106] Meguerdichian S., F. Koushanfar, M. Potkonjak, M. Srivastava, "Coverage Problems in Wireless Ad-Hoc Sensor Networks", in *IEEE Infocom* '2001.

[107] Mesnier, M., M. Wachs, R. Sambasivan, A. Zheng, G. Ganger, "Modeling the Relative Fitness of Storage", in the proceeding of *ACM Sigmetric* '2007.

[108] Min R., A. Chandrakasan, "Top Five Myths about the Energy Consumption of Wireless Communication", poster in the proceeding of ACM Mobicom 2002.

[109] Minsky, Y., A. Trachtenberg, R. Zippel, "Set Reconciliation with Nearly Optimal Communication Complexity", in the proceeding of *IEEE ISIT* '2001.

[110] Minsky, Y., A. Trachtenberg, "Practical Set Reconciliation", in the proceeding of *Allerton* Conference, 2002.

[111] Nauss R.M., "Bond Portfolio Analysis using Integer Programming", in *Financial Optimiz*ation, 1993.

[112] Nielsen, M., "Introduction to Expander Graphs", www.qinfo.org/people/nielsen

[113] Ooi, W., "Dagster: Contributor-Aware End-Host Multicast for Media Streaming in Heterogeneous Environment", in Proc. MMCN, 2005.

[114] Pandurangan, G., P. Raghavan, and E. Upfal, "Building Low-Diameter P2P Networks", in Proc. ACM STOC, 2001.

[115] Pinheiro, E., W. Weber, L. Barroso, "Failure Trends in a Large Disk Drive Population", in the proceeding of *USENIX FAST* '2007.

[116] Plank, J., "A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems", Software Practice and Experience, Sept. 1997.

[117] Plank, J., L. Xu, "Optimizing Cauchy Reed-Solomon Codes for Fault-Tolerant Network Storage Applications", in the proceeding of *IEEE NCA* '2006.

[118] Pourebrahimi, B., K. Bertels, S. Vassiliadis, "A Survey of Peer-to-Peer Networks", Proc. of the 16th Annual Workshop on Circuits, Systems and Signal Processing '2005.

[119] Qiu, T., I. Nikolaidis, F. Li, "On the Design of Incentive-Aware P2P Streaming", in the *Journal of Internet Engineering, Vol. 1, NO. 2*, Oct. 2007.

[120] Rajendran R., D. Rubenstein, "Optimizing the Quality of Scalable Video Streams on P2P Networks", in *Globecom '2004.*

[121] Reed, D. P., "That Sneaky Exponential—Beyond Metcalfe's Law to the Power of Community Building", in Context Magazine, 1999.

[122] Reed, I., G. Solomon, "Polynomial codes over certain finite fields", J. Soc. Indust. Appl. Math., 8(10), pp 300-304, 1960.

[123] Rhea, S., P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, J. Kubiatowicz, "Pond: the OceanStore Prototype", in the proceeding of *USENIX FAST* '2003.

[124] Rodrigues, L., S. Handurukande, J. Pereira, R. Guerraoui, A.-M. Kermarrec, "Adaptive Gossip-Based Broadcast", in Proc. of IEEE DSN '2003.

[125] Rodrigues, R., B. Liskov, "High availability in DHTs: Erasure coding vs replication", in the proceeding of *IPTPS '2005.*

[126] Romberg, J., "Imaging via Compressive Sampling", in *IEEE Signal Processing Magazine*, March 2008.

[127] Rowstron, A., P. Druschel, "Storage Management and Caching in PAST, a Large-scale Persistent Peer-to-Peer Storage Utility", in the proceeding of *ACM SOCP* '2001.

[128] Saroiu, S., K. Gummadi, S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems", in Proc. MMCN, 2002.

[129] Scaglione A., S. D. Serveto, "On the Interdependence of Routing and Data Compression in Multi-hop Sensor Networks", in the Proc. of ACM MobiCom '2002.

[130] Schroeder, B., G. Gibson, "Disk failures in the real world: What does and MTTF of 1,000,000 hours mean to you?", in the proceeding of *USENIX FAST* '2007.

[131] Shakkottai S., S. Srikant, N. Shroff, "Unreliable Sensor Grids: Coverage, Connectivity and Diameter", in *IEEE Infocom '2003.*

[132] Shankar, P., "Expander Codes: The Sipser-Spielman Construction", in journal of Resonance, Jan. 2005.

[133] Sherwood, R., R. Braud, B. Bhattacharjee, "Slurpie: A Cooperative Bulk Data Transfer Protocol", in Proc. IEEE INFOCOM, 2004.

[134] Shokrollahi, A., "Rapter Codes", in IEEE Trans. on Info. Theory, June 2006.

[135] Silber, J., S. Sahu, J. Singh, Z. Liu, "Augmenting Overlay Trees for Failure Resiliency", in Proc. IEEE GLOBECOM, 2004.

[136] Sipser, M., D. Spielman, "Expander Codes", in the proc. of STOC '1996.

[137] Sit E., J. Cates, R. Cox, "A DHT-based Backup System", in the proceeding of the 1st IRIS Student Workshop '2003.

[138] Sloane, N., "The Packing of Spheres", in *Scientific American*, pp. 116-125, January, 1984.

[139] Snavely, N. and Seitz, S. and Szeliski, R., "Photo Tourism: Exploring image collections in 3D", in Proc. ACM SIGGRAPH, 2006.

[140] Sripanidkulchai, K., A. Ganjam, B. Maggs, H. Zhang, "The Feasibility of Supporting Large-Scale Live Streaming Applications with Dynamic Application End-Points", in Proc. ACM SIGCOMM, 2004.

[141] Sripanidkulchai, K., B. Maggs, H. Zhang, "An Analysis of Live Streaming Workloads on the Internet", in Proc. ACM SIGCOMM IMC, 2004.

[142] Stutzbach, D., D. Zappala, R. Rejaie, "The Scalability of Swarming Peer-to-Peer Content Delivery", in Proc. IFIP Networking, 2005.

[143] Sudan, M., "Expander Codes", Lecture Notes, MIT class 6.895, Nov. 2004.

[144] Sung, Y., M. Bishop, S. Rao, "Enabling Contribution Awareness in an Overlay Broadcasting System", in Proc. of ACM SIGCOMM, 2006.

[145] Tati, K., G. Voelker, "On object maintenance in peer-to-peer systems", in the proceeding of *IPTPS*, 2006.

[146] Thommesen, C., "Error-Correcting Capabilities of Concatenated Codes with MDS Outer Codes on Memoryless Channels with Maximum-Likelihood Decoding", in *IEEE Trans. on Info. Theory*, Sept. 1987.

[147] Torralba, A., "Object Recognition and Scene Understanding", MIT Class 6.870, Fall 2008.

[148] Tsiodras, T., http://www.softlab.ntua.gr/~ttsiod /buildWikipediaOffline.html

[149] Tumblin, J., A. Agrawal, R. Raskar, "Why I Want a Gradient Camera", Proc. of IEEE *CVPR '2005.*

[150] Vishnumurthy, V., P. Francis, "On Random Node Selection in P2P and Overlay Networks", Technical Report, Department of Computer Science, Cornell University, 2004.

[151] Wallach, D., "A Survey of Peer-to-Peer Security Issues", in the proceeding of International Symposium on Software Security, 2002.

[152] Wang W., V. Srinivasan , K. Chua, "Trade-offs Between Mobility and Density for Coverage in Wireless Sensor Networks", in *ACM Mobicom* '2007.

[153] Watlington, J., V. M. Bove Jr., "A System for Parallel Media Proceeding", in *Elsevier Journal of Parallel Computing*, 1997.

[154] Weatherspoon, H., J. Kubiatowicz, "Erasure coding vs. replication: a quantitative comparison", in the proceeding of *IPTPS*, 2002.

[155] WiFi standard 802.11n, http://www.broadcom. com/docs/WLAN/802_11n-WP100-R.pdf

[156] Xu, L., J. Bruck, "X-code: MDS array codes with optimal encoding", *IEEE Trans. on Information Theory*, vol. 45, 1999.

[157] Yagi, Y., "Real-time Omnidirectional Image Sensor for Mobile Robot Navigation", in *IEEE ICIC* '2002.

[158] Yang, M., Z. Fei, "A Proactive Approach to Reconstructing Overlay Multicast Trees", in Proc. IEEE INFOCOM, 2004.

[159] Yekhanin, S., "Locally Decodable Codes and Private Information Retrieval Schemes", PhD thesis, MIT, 2007.

[160]Yeung, R., S.-Y. R. Li, N. Cai, and Z. Zhang, "Network Coding Theory", now Publishers, 2005.

[161] Zhang, X., J. Liu, B. Li, T.-S. P. Yum, "DONet/CoolStreaming: A Data-driven Overlay Network for Live Media Streaming", in Proc. IEEE INFOCOM, 2005.

[162] Zhao, Q., "Network Source Coding: Theory and Code Design for Broadcast and Multiple Access Networks", PhD thesis, California Institute of Technology, 2003.

[163] Ziv J., A. Lempel, "A Universal Algorithm for Sequential Data Compression", in *IEEE Trans. on Information Theory*, Vol. IT-23, No. 3, pp. 337~343, May 1977.

[164] "Know Your Enemy: Tracking Botnets", the Honeynet Project and Research Alliance, http://www.honeynet.org, March 2005.

[165] "Spam Report by Microsoft", http://news.bbc.co.uk/1/hi/technology/7988579.stm

[166] J. Graham-Cumming, "Tricks of the Spammer's Trade", in Hakin9 magazine, issue 3, 2004.

[167] S. Biswas, R. Morris, "ExOR: Opportunistic Multi-Hop Routing for Wireless Networks", in the proceeding of ACM SIGCOMM '05, Philadephia, USA, Aug. 2005.

[168] D. P. Reed, "The Third Cloud", MIT Media Lab, 2009.

[169] http://www.flickr.com

[170] http://www.youtube.com

[171] http://www.bittorrent.com

[172] http://en.wikipedia.org/wiki/Redundant_array_of_independent_disks

Appendix A: Proof of Lemma 4.1.9.

In Fig. 4.3.d, we draw several auxiliary lines in order to calculate the area of some sub-regions in the grid unit field. As illustrated in Fig. 4.3.d, the two diagonal lines of the grid unit field intersect at point R. By symmetry, we have $\overline{OR} \perp \overline{XR}$ and the angle $\angle QOR$ is equal to $\arccos(\frac{\sqrt{2}}{2} l/r)$.

Then the angle $\angle POQ$ is equal to $(\frac{\pi}{4} - \arccos(\frac{\sqrt{2}}{2} l/r))$. The area of the region $\overset{\frown}{XWQ}$ is the difference between the area of the triangle ΔXPQ and the area of the region $\overset{\frown}{WPQ}$. The area of the region $\overset{\frown}{XWQ}$ is then equal to $\frac{1}{2} r^2 [\sin(\frac{\pi}{4} - \arccos(\frac{\sqrt{2}}{2} l/r))(1 + \cos(\frac{\pi}{4} - \arccos(\frac{\sqrt{2}}{2} l/r))) \frac{\pi}{4} + \arccos(\frac{\sqrt{2}}{2} l/r)]$.

By the coverage symmetry rule, for a randomly dropped omnidirectional image sensor to the grid field, the probability that its effective view coverage touches one and only one corner with $r < l < \sqrt{2} r$ is the ratio of the sum of the four fishtail-like corner areas and the area of the grid unit field. The sum of the four fishtail-like areas is equal to eight times of the area of the region $\overset{\frown}{XWQ}$. Therefore, the ratio of the sum of the four fishtail-like corner areas and the area of the lattice unit field is then equal to $4 \frac{r^2}{l^2} [\sin(\frac{\pi}{4} - \arccos(\frac{\sqrt{2}}{2} l/r))(1 + \cos(\frac{\pi}{4} - \arccos(\frac{\sqrt{2}}{2} l/r))) \frac{\pi}{4} + \arccos(\frac{\sqrt{2}}{2} l/r)]$. The result follows for N independently-dropped omnidirectional image sensors.

Appendix B: Proof of Lemma 4.1.10.

As illustrated in Fig. 4.3.d, we draw the line through point S and point R, which intersects with the grid edge $\overline{XO}$ at point T. The angle $\angle TOS$ is equal to $\arccos(\frac{1}{2}l/r)$. Then the angle $\angle ROS$ is equal to $(\arccos(\frac{1}{2}l/r)-\frac{\pi}{4})$. The area of the region $\overset{\frown}{VRS}$ is $\frac{r^2}{2}(\arccos(\frac{1}{2}l/r)-\frac{\pi}{4})$ -$\frac{1}{4}l$ $(r\sin(\arccos(\frac{1}{2}l/r))-l/2)$. By symmetry, the area of the region $\overset{\diamond}{LJIS}$ is eight times of the area of the region of $\overset{\frown}{VRS}$. The area of the region $\overset{\diamond}{LJIS}$ is then equal to $2(2r^2(\arccos(\frac{1}{2}l/r)-\frac{\pi}{4})$ -l $(r\sin(\arccos(\frac{1}{2}l/r))-l/2))$.

By the coverage symmetry rule, for a randomly dropped omnidirectional image sensor to the grid field, the probability that its effective view coverage touches exactly four corners with $r<l<\sqrt{2}r$ is the ratio of the area of the region $\overset{\diamond}{LJIS}$ and the area of the grid unit field. That is $2(2r^2(\arccos(\frac{1}{2}l/r)-\frac{\pi}{4})$ -l $(r\sin(\arccos(\frac{1}{2}l/r))-l/2))/l^2$. The result follows for N independently-dropped omnidirectional image sensors.

Appendix C: Proof of Lemma 4.1.11.

As illustrated in Fig. 4.3.d, the two diagonal lines of the grid unit intersect at point R. The angle $\angle QOR$ is equal to $\arccos(\frac{\sqrt{2}}{2} l / r)$. Then the angle $\angle QOU$ is equal to $2\arccos(\frac{\sqrt{2}}{2} l / r)$. The sum of the area of the regions of $QL\overset{\diamond}{S}UIJ$ and $KSI\overset{\diamond}{M}JL$ is equal to $4(r^2 \arccos(\frac{\sqrt{2}}{2} l / r) - \frac{\sqrt{2}}{2} l\ r \sin(\arccos\ (\frac{\sqrt{2}}{2} l / r)))$.

By the coverage symmetry rule, for a randomly dropped omnidirectional image sensor to the grid field, the probability that its effective view coverage touches exactly three corners with $r < l < \sqrt{2}r$ is the ratio of the sum of the areas of $\overset{\frown}{QJL}$, $\overset{\frown}{KLS}$, $\overset{\frown}{USI}$ and $\overset{\frown}{MIJ}$ and the area of the grid unit field, which is $(4(r^2 \arccos(\frac{\sqrt{2}}{2} l / r) - \frac{\sqrt{2}}{2} l\ r \sin(\arccos\ (\frac{\sqrt{2}}{2} l / r)))4r\,(2r\,(\arccos(\frac{1}{2} l / r) - \frac{\pi}{4})\sqrt{2}l \sin(\arccos\ (\frac{1}{2} l / r) - \frac{\pi}{4})))/ l^2$. The result follows for N independently-dropped omnidirectional image sensors.

Appendix D: Proof of Lemma 4.1.13.

As shown in Fig. 4.3.e, we draw several auxiliary lines in order to calculate the area of some regions of interest in the grid unit field. The area of the region $\overset{\frown}{VRS}$ is equal to $\frac{r^2}{2}(\arccos(\frac{l/2}{r})-\frac{\pi}{4})-\frac{l}{4}(r\sin(\arccos(\frac{l/2}{r}))-l/2)$. By symmetry, the area of the region $\overset{\diamond}{PSQU}$ is eight times of the area of the region of $\overset{\frown}{VRS}$. The area of the region $\overset{\diamond}{PSQU}$ is then $4r^2(\arccos(\frac{l/2}{r})-\frac{\pi}{4})-2l(r\sin(\arccos(\frac{l/2}{r}))-l/2)$.

By the coverage symmetry rule, if an omnidirectional image sensor is dropped in the region of$\overset{\diamond}{PSQU}$, it will cover at least four corners. The probability that a randomly dropped omnidirectional image sensor covers at least four corners is the ratio of the area of the region of $\overset{\diamond}{PSQU}$ and the area of the grid unit field. That is $(4r^2(\arccos(\frac{l/2}{r})-\frac{\pi}{4})-2l(r\sin(\arccos(\frac{l/2}{r}))-l/2))/l^2$. The result follows for N independently-dropped omnidirectional image sensors.

Appendix *E*: The packing of spheres.

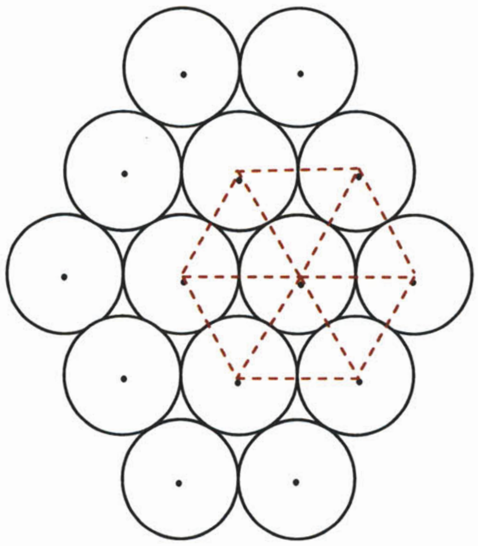


Figure 4.10: The packing of spheres.

Appendix *F*: The approximation used in Eq. (4.15).

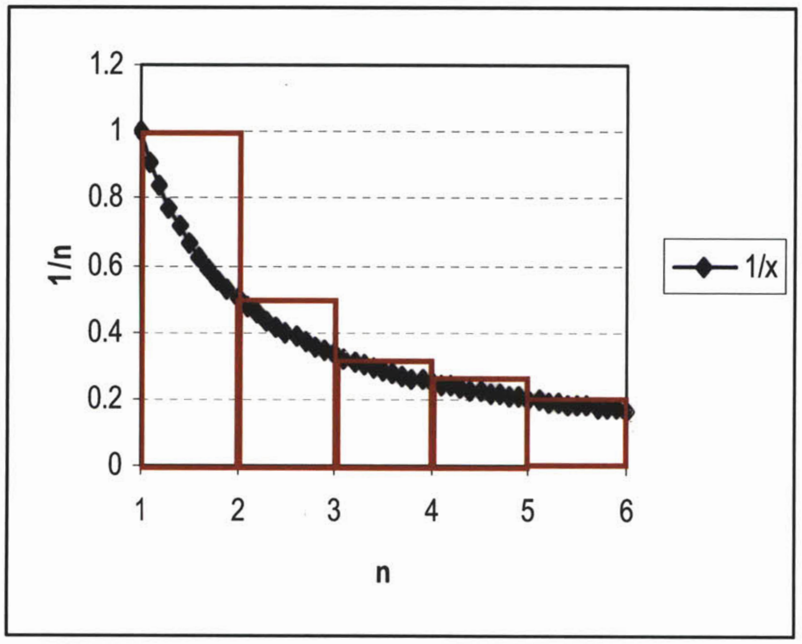


Figure 4.11: An illustration of the staircase approximation of $\sum_{i=1}^{n}\frac{1}{i} \approx \int_{1}^{n}\frac{1}{x}dx$.