Coding, Computing, and Communication in Distributed Storage Systems

MAJID GERAMI

Doctoral Thesis in Electrical Engineering
Stockholm, Sweden 2016
To my mother and father!
Abstract

Conventional studies in communication networks mostly focus on securely and reliably transmitting data from a source node (or multiple source nodes) to multiple destinations. A more general problem appears when the destination nodes are interested in obtaining functions of the data available in distributed source nodes. For obtaining a function, transmitting all the data to a destination node and then computing the function might be inefficient. In order to exploit the network resources efficiently, the general problem offers distributed computing in combination with coding and communication. This problem has applications in distributed systems, e.g., in wireless sensor networks, in distributed storage systems, and in distributed computing systems. Following this general problem formulation, we study the optimal and secure recovery of the lost data in storage nodes and in reconstructing a version of a file in distributed storage systems.

The significance of this study is due to the fact that the new trends in communications including big data, Internet of things, low latency, and high reliability communications challenge the existing centralized data storage systems. Distributed storage systems can rectify those issues by distributing thousands of storage nodes (possibly around the globe), and then benefiting users by bringing data to their proximity. Yet, distributing the storage nodes brings new challenges. In these distributed systems, where storage nodes are connected through links and servers, communication plays a main role in their performance. In addition, a part of network may fail or due to communication failure or delay there might exist multi versions of a file. Moreover, an intruder can overhear the communications between storage nodes and obtain some information about the stored data. Therefore, there are challenges on reliability, security, availability, and consistency.

To increase reliability, systems need to store redundant data in storage nodes and employ error control codes. To maintain the reliability in a dynamic environment where storage nodes can fail, the system should have an autonomous repair process. Namely, it should regenerate the failed nodes by the help of other storage nodes. The repair process demands bandwidth, energy, or in general transmission costs. We propose novel techniques to reduce the repair cost in distributed storage systems.

First, we propose surviving nodes cooperation in repair, meaning that surviving nodes can combine their received data with their own stored data and then transmit toward the new node. In addition, we study the repair problem in multi-hop networks and consider the cost of transmitting data between storage nodes. While
classical repair model assumes the availability of direct links between the new node and surviving nodes, we consider that such links may not be available either due to failure or their costs. We formulate an optimization problem to minimize the repair cost and compare two systems, namely with and without surviving nodes cooperation.

Second, we study the repair problem where the links between storage nodes are lossy e.g., due to server congestion, load balancing, or unreliable physical layer (wireless links). We model the lossy links by packet erasure channels and then derive the fundamental bandwidth-storage tradeoff in packet erasure networks. In addition, we propose dedicated-for-repair storage nodes to reduce the repair-bandwidth.

Third, we generalize the repair model by proposing the concept of partial repair. That is, storage nodes may lose parts of their stored data. Then in partial repair, the lost data is recovered by exchanging data between storage nodes and using the available data in storage nodes as side information. For efficient partial-repair, we propose two-layer coding in distributed storage systems and then we derive the optimal bandwidth in partial repair.

Fourth, we study security in distributed storage systems. We investigate security in partial repair. In particular, we propose codes that make the partial repair secure in the senses of strong and weak information-theoretic security definitions.

Finally, we study consistency in distributed storage systems. Consistency means that distinct users obtain the latest version of a file in a system that stores multi versions of a file. Given the probability of receiving a version by a storage node and the constraint on the node storage space, we aim to find the optimal encoding of multi versions of a file that maximizes the probability of obtaining the latest version of a file or a version close to the latest version by a read client that connects to a number of storage nodes.
Sammanfattning


Även om det främst antas att alla länkar mellan lagringssnoder är felfria, studier

För det tredje, vi generaliserar reparationsmodellen genom att föreslå partiell reparation. Vi modellerar partiell nodfel, vilket innebär att lagringsnoder i vår modell kan förlopa delar av sitt lagrade data. Sedan i partiell reparation, återvinns det förlopa data genom att utbyta data mellan lagringsnoder. Tillgång till sidoinformation i partiell reparation gör det till ett specifikt problem att studera. Vi studerar partiell reparation i trådbundna distribuerade lagringsystem och även i trådlös caching med trådlösa sändningskanaler.

För det fjärde, studerar vi säkerhet i distribuerade lagringssystem. När lagringsnoder är fördelade runt om i världen, och speciellt när Internet används för kommunikation mellan lagringsnoder, kan en avlyssnare höra överförda data mellan lagringsnoder och får på så sätt viss information om lagrat data. Vi undersöker informationsteoretisk säkerhet i partiell reparation.

Framför allt gör vi den partiella reparationen säker i avseendet svag- och stark informationsteoretisk säkerhet.

Slutligen studerar vi korrektheten av läs- och skrivprocesser i distribuerade lagringssystem.
Acknowledgments

I would like to take this opportunity to acknowledge all those who have supported me in the development of this thesis.

First and foremost, I would like to express my sincere gratitude to Prof. Ming Xiao. Ming gave me this opportunity to become a Ph.D. student at Communication Theory Department of KTH. He introduced me the concept of network coding, the topic that I have been interested in since our first meeting. Besides, I have learned a lot from Ming, from dealing with reviewers and how to accurately respond to the reviewer comments, and to many invaluable points in life. I am very thankful to him for invaluable discussions, insightful suggestions and feedbacks, and supports through my years of study. Also, I want to thank Prof. Mikael Skoglund for accepting me as a Ph.D. student in Communication Theory Department and for being my co-advisor during my study. Mikael has been always supportive and I have had the chance to receive invaluable comments from him.

I would like to thank Prof. Mohammad Ali Maddah Ali and Prof. Reinaldo Valenzuela for accepting me as a visiting researcher at Wireless Group of Nokia Bell Labs, New Jersey, USA. This opportunity has broadened my knowledge and my view on research tremendously. Mohammad Ali generously gave me lots of his time and he taught me invaluable techniques in research. I am also grateful for all friends in Bell Labs who made my life enjoyable and rewarding. In particular, I would like to thank Prof. Murali Kodialam, Prof. Dmitry Chizhik and Dr. Jinfeng Du for the fruitful discussions we had. I would also like to acknowledge Ericsson and Stiftelsen Engbloms Stipendiefond for supporting this visit.

I would like to thank Prof. Camilla Hollanti for acting as the opponent for this thesis. I also thank the grading committee formed by Prof. Daniel Enrique Lucani Roetter, Prof. Alexandre Graell i Amat, Prof. Jim Dowling, and Prof. Magnus Jansson. I am thankful to Prof. Markus Flierl for the quality review of this thesis. I am also thankful to Prof. Lars Kildehøj for acting as the session chairman in my Ph.D. defense.

During my Ph.D. study, I had chances to collaborate with Prof. Mohammad Ali Maddah Ali, Dr. Somayeh Salimi, Prof. Carlo Fischione, Prof. Panagiotis Papadimitratos, Dr. Kenneth Shum, and Awassada Phatthum. I would like to thank all of them for the fruitful collaboration and for invaluable feedbacks and suggestions.

I am indebted to Prof. Sarah Johnson and Dr. Lawrence Ong for their generosity in hosting me at School of Electrical Engineering and Computer Science in New
I am thankful to Prof. Camilla Hollanti for her generosity in hosting me at Department of Mathematics and Systems Analysis in Aalto University. I am also thankful to Ph.D. fellow Joonas Pääkkönen at that department. We had great discussions about new problems in distributed storage systems. I am also thankful to Prof. Camilla Hollanti and Prof. Dejan Vukobratovic for inviting me to the COST meeting at University of Novi Sad.

I am also very thankful to Prof. Mikael Gidlund for giving me the internship opportunity at ABB Corporate Research. The project of indoor localization was very interesting and I learned many things there. I had very skillful team members that I have learned many things from them. In particular, I had very fantastic discussions with Dr. Ali Zaidi, Dr. Johan Sjöberg, Thomas Fuglsang, Anders Eslkildsen, and Mikaela Ahlén.

During my PhD study, I received great comments and suggestions from Dr. Majid Nasiri Khormooji, Dr. Hamed Farhadi, Dr. Ali Zaidi, Dr. Serveh Shalmashi, Farshad Naghibi, Dr. Somayeh Salimi, Dr. Jinfeng Du, and Dr. Efthymios Statthakis. Thanks all of you!

I am also thankful to my teachers at KTH (The Royal Institute of Technology), as well as my teachers at Sharif University of Technology, and my teachers at Ferdowsi University of Mashahad.

I am sincerely grateful to Guang Yang, Aalla Tarighati, Hoessein Shokri, Farshad Naghibi, Ahmad Zaki, Due Liu, Nima Najjar Moghadam, Ehsan Olfa, and Hadi Ghauq for proofreading different parts of this thesis. I am also thankful to Peter Larsson for his kind helps in editing the Swedish parts of this thesis. Additionally I want to thank all my colleagues and friends from the Communication Theory and Signal Processing labs for providing a great working environment. It was so nice to share the office with Ahmad Zaki all these years. Zaki is one of the most knowledgable persons, and his smart ideas have been always fantastic. I would like to thank Raine Tiivel, Irene Kindblom, Tove Schwartz, and Cecilia Forssman for administrative support. I also thank the computer support group, in particular Pontus Friberg and Niclas Hornoy, for providing reliable resources.

I would like to thank all my friends in Stockholm who make my life enjoyable and memorable. Special thanks to Kiomars, Maryam, Mohsen, Hengameh, Helena, Rolf, Aalla, Tahereh, Amirpasha, Nafiseh, Farshad, Serveh, Arash, Somayeh, Mehdi, Majid, Hamed, Maryam, Altamash Khan, and Ali. I would like to thanks Green Race for Sustainability group, our bicycle runners club, including Bruce, Nan Qi, Liyun, and Sebastian.

Finally, I would like to express my endless gratitude to my family. I want to thank Mahsa for all her love, for all great moments we shared over the last years, and for all her support and encouragement during the difficult times. Thank you for making me happy every day. I would like to thank my brothers and my sisters, my mother-in-law (Mahvash), my grandmother (Madar joon), my aunts, and my uncles. Special thanks to Mehdi, Mehran, Gity, Behrooz, Piruz, Mohsen, Najma,
Somayeh, Behjat, Mehrdad, and Aboulfazl for their support and encouragement. I would like to especially thank my mother and my father for their love, care and encouragement. This thesis is dedicated to you with love!

Majid Gerami
Stockholm, October 2016
Contents

Abstract v
Sammanfattning vii
Acknowledgments ix
Contents xii
Notation xvii
List of Acronyms xix
List of Figures xxi

1 Introduction 1
1.1 Coding in Distributed Storage Systems . . . . . . . . . . . . . . . . 5
1.2 Thesis Scope and Contributions . . . . . . . . . . . . . . . . . . . . 6
1.3 Copyright Notice . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

2 Background 11
2.1 Entropy and Mutual Information . . . . . . . . . . . . . . . . . . . . 11
2.2 Binary Erasure Channel . . . . . . . . . . . . . . . . . . . . . . . . . 12
2.3 Packet Erasure Channel . . . . . . . . . . . . . . . . . . . . . . . . . 13
2.4 Network Coding . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
2.5 Secure Network Coding . . . . . . . . . . . . . . . . . . . . . . . . . 15
2.5.1 Wiretap Channel Type II . . . . . . . . . . . . . . . . . . . . . . 15
2.5.2 Strong Security Over Wiretap Network II . . . . . . . . . . . . 17
2.5.3 Weak Security Over Wiretap Network II . . . . . . . . . . . . . 18
2.6 Coding in Storage Systems . . . . . . . . . . . . . . . . . . . . . . . 18
2.6.1 Regenerating codes . . . . . . . . . . . . . . . . . . . . . . . . . 20
2.7 Consistency in Distributed Storage Systems . . . . . . . . . . . . . . 22
2.7.1 Coding in Consistent Distributed Storage Systems . . . . . . . 23
### I Surviving Nodes Cooperation in Repair

#### 3 Optimal-cost Repair with Surviving Nodes Cooperation

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>27</td>
</tr>
<tr>
<td>3.2</td>
<td>Motivating Examples</td>
<td>29</td>
</tr>
<tr>
<td>3.3</td>
<td>Problem Formulation</td>
<td></td>
</tr>
<tr>
<td>3.3.1</td>
<td>System Model</td>
<td>33</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Repair-cost Formulation</td>
<td>34</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Complexity of the proposed algorithm</td>
<td>38</td>
</tr>
<tr>
<td>3.4</td>
<td>Achievable Codes for Minimum Repair-cost</td>
<td>39</td>
</tr>
<tr>
<td>3.5</td>
<td>Numerical Results</td>
<td></td>
</tr>
<tr>
<td>3.5.1</td>
<td>Large Scale Tandem Storage Network</td>
<td>41</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Large Scale Grid Storage Network</td>
<td>43</td>
</tr>
<tr>
<td>3.5.3</td>
<td>Fully-Connected Storage Network</td>
<td>43</td>
</tr>
<tr>
<td>3.6</td>
<td>Explicit Construction of Exact Optimal-cost Codes</td>
<td></td>
</tr>
<tr>
<td>3.6.1</td>
<td>Explicit Code Construction in Tandem Networks</td>
<td>45</td>
</tr>
<tr>
<td>3.6.2</td>
<td>Explicit Code Construction in Grid Networks</td>
<td>47</td>
</tr>
<tr>
<td>3.7</td>
<td>Conclusions</td>
<td>51</td>
</tr>
<tr>
<td>3.8</td>
<td>Appendices</td>
<td></td>
</tr>
<tr>
<td>3.8.1</td>
<td>Proof of Proposition 3.2</td>
<td>51</td>
</tr>
<tr>
<td>3.8.2</td>
<td>Proof of Corollary 3.1</td>
<td>52</td>
</tr>
</tbody>
</table>

### II Distributed Storage Systems with Packet Erasure Channels

#### 4 Repair for DSSs with Packet Erasure Channels

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>57</td>
</tr>
<tr>
<td>4.2</td>
<td>Background and Related Works</td>
<td>59</td>
</tr>
<tr>
<td>4.3</td>
<td>System Model</td>
<td>61</td>
</tr>
<tr>
<td>4.4</td>
<td>Asymptotic Analysis</td>
<td></td>
</tr>
<tr>
<td>4.4.1</td>
<td>Information Flow Graph</td>
<td>63</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Optimal Storage-Bandwidth Tradeoff in Erasure Networks</td>
<td>64</td>
</tr>
<tr>
<td>4.5</td>
<td>Reducing Repair-Bandwidth by DR Storage Nodes</td>
<td></td>
</tr>
<tr>
<td>4.5.1</td>
<td>Motivation</td>
<td>66</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Minimum DR Storage Capacity for the EMSR Codes</td>
<td>68</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Minimum Repairing Storage Capacity for EMBR Codes</td>
<td>70</td>
</tr>
<tr>
<td>4.5.4</td>
<td>Repairing a DR Storage Node</td>
<td>71</td>
</tr>
<tr>
<td>4.5.5</td>
<td>DR Storage Nodes for non-EMSR and non-EMBR Codes</td>
<td>73</td>
</tr>
<tr>
<td>4.5.6</td>
<td>Repair with DR Storage Nodes in Packet Erasure Networks</td>
<td>74</td>
</tr>
<tr>
<td>4.6</td>
<td>Finite File Size Analysis</td>
<td></td>
</tr>
<tr>
<td>4.6.1</td>
<td>The Probability of Successful Repair</td>
<td>75</td>
</tr>
<tr>
<td>4.6.2</td>
<td>Practical-Repair-Bandwidth</td>
<td>76</td>
</tr>
<tr>
<td>4.6.3</td>
<td>Reducing Practical-Repair-Bandwidth</td>
<td>77</td>
</tr>
</tbody>
</table>
Notation

- \( p(x) \) Probability density function of a random variable \( X \)
- \( p(x|y) \) Conditional probability density function of \( X \) given \( Y \)
- \( H(X) \) Entropy of a random variable \( X \)
- \( H(X|Y) \) Conditional entropy of \( X \) given \( Y \)
- \( I(X;Y) \) Mutual information between \( X \) and \( Y \)
- \( \mathbf{A} \) Matrix \( \mathbf{A} \)
- \( \mathbf{a} \) Vector \( \mathbf{a} \)
- \( \mathcal{A} \) Set \( \mathcal{A} \)
- \( \mathbf{A}^T \) Transpose of matrix \( \mathbf{A} \)
- \( |\mathcal{A}| \) Cardinality of set \( \mathcal{A} \)
- \( [n] \) Set of integers between \( 1 \) and \( n \), i.e., \( [n] = \{1, 2, \ldots, n\} \)
- \( \text{det}(\mathbf{A}) \) Determinant of a matrix \( \mathbf{A} \)
- \( GF(q) \) The finite field of size \( q \)
List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNC</td>
<td>Surviving node cooperation</td>
</tr>
<tr>
<td>DR</td>
<td>Dedicated-for-repair</td>
</tr>
<tr>
<td>P2P</td>
<td>Peer-to-peer</td>
</tr>
<tr>
<td>MDS</td>
<td>Maximum distance separable</td>
</tr>
<tr>
<td>MSR</td>
<td>Minimum storage regenerating</td>
</tr>
<tr>
<td>MBR</td>
<td>Minimum bandwidth regenerating</td>
</tr>
<tr>
<td>EMSR</td>
<td>Extended MSR</td>
</tr>
<tr>
<td>EMBR</td>
<td>Extended MBR</td>
</tr>
<tr>
<td>DHT</td>
<td>Distributed Hash Table</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>CRC</td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td>i.i.d.</td>
<td>Independent and identically distributed</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>1.1</td>
<td>A typical structure for a geographically distributed storage system</td>
</tr>
<tr>
<td>1.2</td>
<td>A typical data-center architecture</td>
</tr>
<tr>
<td>1.3</td>
<td>A typical peer-to-peer network with five nodes</td>
</tr>
<tr>
<td>1.4</td>
<td>A typical distributed computing system with five nodes</td>
</tr>
<tr>
<td>1.5</td>
<td>An application for delay tolerant networks</td>
</tr>
<tr>
<td>1.6</td>
<td>A typical wireless sensor network</td>
</tr>
<tr>
<td>2.1</td>
<td>A binary erasure channel</td>
</tr>
<tr>
<td>2.2</td>
<td>The butterfly network</td>
</tr>
<tr>
<td>2.3</td>
<td>The wiretap channel type II</td>
</tr>
<tr>
<td>2.4</td>
<td>Data security in the butterfly network</td>
</tr>
<tr>
<td>2.5</td>
<td>The optimal bandwidth-storage tradeoff</td>
</tr>
<tr>
<td>3.1</td>
<td>Surviving node cooperation can reduce the repair cost</td>
</tr>
<tr>
<td>3.2</td>
<td>A distributed storage system in a 4-node tandem network</td>
</tr>
<tr>
<td>3.3</td>
<td>The information flow graph in the classical repair model</td>
</tr>
<tr>
<td>3.4</td>
<td>Regenerating by surviving node cooperation in a tandem network</td>
</tr>
<tr>
<td>3.5</td>
<td>The information flow graph for a tandem network</td>
</tr>
<tr>
<td>3.6</td>
<td>Repair in a four-node tandem network</td>
</tr>
<tr>
<td>3.7</td>
<td>A large scale tandem storage network</td>
</tr>
<tr>
<td>3.8</td>
<td>A large scale grid storage network</td>
</tr>
<tr>
<td>3.9</td>
<td>Repair-cost comparison between the proposed scheme and classical repair</td>
</tr>
<tr>
<td>3.10</td>
<td>Repair-cost comparison between the proposed scheme and classical repair</td>
</tr>
<tr>
<td>3.11</td>
<td>Exact and optimal-cost repair in the 2 × 3 grid network</td>
</tr>
<tr>
<td>3.12</td>
<td>Cut analysis in a tandem network</td>
</tr>
<tr>
<td>3.13</td>
<td>Cut analysis in a 2 × 3 grid network</td>
</tr>
<tr>
<td>4.1</td>
<td>Information flow graph for distributed storage systems</td>
</tr>
<tr>
<td>4.2</td>
<td>Fundamental tradeoff with different packet erasure probability</td>
</tr>
<tr>
<td>4.3</td>
<td>A DR storage node helps the repair of nodes 1 for an MSR code</td>
</tr>
<tr>
<td>4.4</td>
<td>The DR storage node helps the repair of nodes 2 for an MSR code</td>
</tr>
<tr>
<td>4.5</td>
<td>The DR storage node helps the repair of nodes 3 for an MSR code</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>4.6</td>
<td>A DR storage node helps the repair of nodes 1 for an MBR code</td>
</tr>
<tr>
<td>4.7</td>
<td>The DR storage node helps the repair of nodes 2 for an MBR code</td>
</tr>
<tr>
<td>4.8</td>
<td>The DR storage node helps the repair of nodes 3 for an MBR code</td>
</tr>
<tr>
<td>4.9</td>
<td>The DR storage node helps the repair of nodes 4 for an MBR code</td>
</tr>
<tr>
<td>4.10</td>
<td>Repair of the DR storage node for EMSR codes</td>
</tr>
<tr>
<td>4.11</td>
<td>Repair of the DR storage node for EMBR codes</td>
</tr>
<tr>
<td>4.12</td>
<td>$P_{4}$ for different values of subpacketization and bandwidth overhead ratio</td>
</tr>
<tr>
<td>4.13</td>
<td>Probability of successful repair</td>
</tr>
<tr>
<td>4.14</td>
<td>Optimal values for $d_1$ and $d_2$</td>
</tr>
<tr>
<td>5.1</td>
<td>Two-layer coding in distributed storage systems</td>
</tr>
<tr>
<td>5.2</td>
<td>Information flow graph in partial repair</td>
</tr>
<tr>
<td>5.3</td>
<td>Bandwidth-storage tradeoff in partial-repair</td>
</tr>
<tr>
<td>5.4</td>
<td>Information flow graph for partial repair</td>
</tr>
<tr>
<td>6.1</td>
<td>Partial repair in distributed storage systems</td>
</tr>
<tr>
<td>6.2</td>
<td>Secure partial repair in distributed storage systems</td>
</tr>
<tr>
<td>6.3</td>
<td>Deriving the optimal partial repair</td>
</tr>
<tr>
<td>6.4</td>
<td>Secure MDS encoding</td>
</tr>
<tr>
<td>6.5</td>
<td>Secure and exact partial repair in Example 2</td>
</tr>
<tr>
<td>6.6</td>
<td>The effect of fragment erasure pattern</td>
</tr>
<tr>
<td>6.7</td>
<td>Comparing random network coding with our deterministic optimal codes</td>
</tr>
<tr>
<td>6.8</td>
<td>Information flow graph for the multiple faulty nodes</td>
</tr>
<tr>
<td>6.9</td>
<td>Cut analysis in information flow graph for the multiple faulty nodes</td>
</tr>
<tr>
<td>7.1</td>
<td>The system model in consistency problem</td>
</tr>
<tr>
<td>7.2</td>
<td>State matrix</td>
</tr>
<tr>
<td>7.3</td>
<td>Remove-and-add process</td>
</tr>
<tr>
<td>7.4</td>
<td>Remove-and-add process for small $d$</td>
</tr>
<tr>
<td>7.5</td>
<td>Remove-and-add graph for large $d$</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Data storage is becoming an important part of communication systems and the other way around, communication between storage units plays an important role on performance of storage systems. Moreover, communication will be the main challenge for future distributed computing systems [L Maya16, TLR12]. All these emphasize the significance of the interrelation between storage, computing and communication in distributed systems. We study this interrelation to find fundamental limits or tradeoffs in distributed storage systems and the role of coding in these systems. In our study the notion of distributed storage system is rather broad and it consists of all networks having nodes with data storage, including data-centers, peer-to-peer (P2P) networks, distributed cloud storage networks, distributed computing systems, wireless caching networks, delay tolerant networks, and wireless sensor networks.

The study on distributed storage systems has attracted considerable research attentions recently. One reason is that the volume of generated data in the world has been significantly growing, such that it has drawn a lot of attention to big data and big data explosion [Int]. To clarify the situation, consider that in 2015, Facebook users shared 2.5 million files per minute, Instagram users uploaded 220,000 photos per minute, Youtube users uploaded every minute 72 hours of new video, and Email users sent nearly 200 millions posts every minute. Moreover, Google analysed 20 petabytes of data every day, while all the data generated by human being in history till 2013 was 50 petabytes. In addition, all these figures are expected to increase by a multiplicative factor of 6 till 2020 [Int]. Thus, providing enough storage space is challenging.

Distributed storage systems are the key ingredients in fulfilling the increasing demand for data storage. Affording this large data volume in a centralized fashion requires very high technology in memory manufacturing which makes the system very expensive. In addition, the centralized storage unit is not scalable. That is, by increasing the volume of data a new storage device with a larger capacity must be replaced. The problems of centralized storage systems can be solved by using multiple cheap storage units that are connected together through a network, leading
to a distributed storage system. In distributed storage systems which include distributed cloud storage systems, P2P cloud storage systems and private/public data centers, users can store, archive, or back up their data on the (geographically) distributed storage nodes. Dropbox [DMMM+12], Google File Systems [GGL03], and AmazonS3 [PIRG08] are among the examples of these storage systems. Figure 1.1 shows a typical structure of a geographically distributed storage system. The system contains multiple data-centers across the world. A user connects to the closest data-center among the available data-centers and obtains its requested data. It is useful to know that each data-center itself contains hundreds of storage racks, and each rack contains multiple storage nodes as shown in Figure 1.2. Communication between storage units in a rack is coordinated by the rack’s server. Communication between storage units in different racks is coordinated through multiple servers which are connected through a hierarchical network structure [AFLV08]. This implies that transmission costs between different storage nodes might be different.

Another application for distributed storage systems is P2P networks. In P2P
networks, a number of computing nodes share their resources. In these networks, there is not a central control unit which coordinates communication between nodes or monitors the presence or absence of a node. Computing nodes can easily leave or join a network. In addition, there is not a fixed client-server relation between nodes, yet a node can act simultaneously as a client of a service, or as a server to provide services to another user. Figure 1.3 shows a typical P2P network consisting of five computing nodes. A user can send a file request in these systems mainly by the two following algorithms: (1) a client sends its request of a file to all its neighbouring nodes through a flooding algorithm, (2) a client sends its request to a list of neighboring nodes determined by a table, denoted as distributed hash table (DHT). In this thesis, we do not study these algorithms and interested readers are referred to [DLS+04, RL05] and references therein.

Distributed computing systems are also among the applications of distributed storage systems. In these systems, as shown in Fig. 1.4, there are a number of computing nodes having their own individual memory units. Distributed computing nodes exchange information to achieve the system goals. It may worth to note that there is a difference between distributed computing systems with parallel computing systems. The difference is due to the fact that in distributed computing systems each computing node has its own private memory while in parallel commuting systems different computing nodes share one memory.

Another reason for the significance of distributed storage systems is that data storage is becoming an indispensable part of communication systems, e.g., in wire-
Introduction

In distributed computing systems, distributed and distributed memory networks, wireless sensor networks and Internet of Things (IoTs). Storing data in proximity of users is typically denoted as caching. In wireless caching networks, storing parts of popular files in storage units of user terminals has shown considerable reduction in transmission load [MAN14]. Data storage is also beneficial for wireless caching networks in reducing delay, energy, and in general transmission costs, especially where mobile storage nodes are capable of device-to-device communication [GSD*12, WYC96, JCMH16, GXS15, SGD*13, NSG03, PHT13, PHT15, PBHT16, SMST3, Sha14]. Availability of high capacity data-storage space in user equipment motivates the use of equipment’s storage space to reduce the base station’s transmission load during busy hours. In these systems, in off-peak hours, parts of the most popular files stored in user equipment. During busy hours, a part of user file requests can be served by the local storage units.

Among applications of communication networks with data storage nodes are delay tolerant networks and wireless sensor networks. In delay tolerant networks, users can tolerate a bounded delay in receiving a content. An application of delay tolerant networks over a wireless network has been shown in Figure 1.5. In this figure, a base station broadcasts a file to the mobile stations which individually have a limited storage space. Each node stores a part of the file. A client, can obtain the file by connecting to a number of mobile storage nodes, even though the client might be out of base station’s coverage. In another application, in wireless sensor networks, as shown in Figure 1.6 measured data is stored in redundant storage nodes to increase reliability.

Figure 1.4: A typical distributed computing system with five nodes.
Considering the above applications, we can say in summary that distributing storage nodes benefit the systems in scalability, availability and reliability. Scalability means that the system performs the same after increasing users or increasing demands for data storage. For instance, new storage nodes can be easily added to the system without affecting the work of other storage nodes. Availability means that a user receives a proper response in a decent time, even if some servers or links fail. And finally, reliability means that the stored data in the system is still accessible even if some storage nodes fail or parts of stored data is lost.

1.1 Coding in Distributed Storage Systems

Traditionally, to have a high reliability in distributed storage systems, a copy of a file is replicated in several distinct storage nodes. Then, if a copy is lost, there exists at least a copy of the stored file. Yet, replication does not exploit the given redundancy efficiently. In a more general setting, one can use coding in distributed storage systems. While most of the existing distributed storage systems use replication for reliability of their hot data (highly requested data), recently coding has been suggested for storing cold data (archival data) in large-scale distributed storage systems, e.g., in Google File System [FLP+10], Hadoop FS [TSA+10], Microsoft Azure [HSX+12], and Wuala P2P networks [MPA+11]. As a node failure can be modeled as an erasure, erasure codes can be used in these systems. In particular, if a source file of size $M$ is divided into $k$ parts and encoded to $n$ parts such that any $k$ parts can reconstruct the source file, then this code is optimal in the use of redundancy for providing reliability. These codes are termed as maximum distance separable (MDS) codes.

However, the above advantage of coding does not come free and coding compared to replication may impose higher costs to the storage systems in some scenarios such as repair. Recently, the costs in repair has been studied from different aspects, e.g., in repair bandwidth [DGW+10], the number of disk I/O reads [ERR10], and repair
A new class of erasure codes, namely regenerating codes based on network coding \cite{ACLY00,KM03}, are proposed in \cite{DGW10,Wu10a}. In the proposed codes, the new node may not have the same encoded data as the failed node; however, the new node and the surviving nodes still preserve the property that a fixed number of nodes can reconstruct the original file. This kind of repair is termed as functional repair. The exact regeneration of a new node has been studied in \cite{RSKR09a,SRKR12}. Since the proposed approach in \cite{DGW10} uses network coding to achieve the optimal repair-bandwidth codes, it requires multiple reads from any storage node. To reduce the number of reads, fractional repetition codes have been introduced by El Rouayheb and Ramchandran in \cite{ERR10}. Another important criterion is the number of surviving nodes that are connected in repair, denoted as the repair locality. The repair locality has been studied by Papailiopoulos and Dimakis in \cite{PD14}. Complexity in repair has been studied in \cite{KIAAB15} and erasure codes having low repair bandwidth and low complexity has been proposed. We study the transmission cost of repair in a distributed storage system whose nodes are connected through an arbitrary network topology and network links have different costs. Next, we study the erasure codes for distributed storage systems where not only the storage nodes but also the links between storage nodes face failure. We present how much extra repair bandwidth must be transmitted due to packet erasures on the links. We then propose DR storage nodes to reduce the repair bandwidth. Later, we generalize the repair problem by introducing a more general model for node failure, denoted as partial node failure. We study repair in these networks. We investigate the security of repair if an eavesdropper overhears some repairing packets. Finally, we investigate consistency in distributed storage systems.

1.2 Thesis Scope and Contributions

In this thesis, we study the role of coding in distributed storage system from different aspects. The thesis has four parts. Each part consists of one or two chapters. In Part I, we study the repair problem in multi-hop networks. In Part II, we investigate the
repair problem in packet erasure networks. In Part III, we study partial repair and the security in partial repair in distributed storage systems. Finally in Part IV, we study consistency in distributed storage systems.

Some of the results presented in the thesis have already been published in journals and conferences, and some are under review. Parts of the thesis are adopted from the corresponding research papers nearly verbatim. In the following we give a brief introduction of each chapter along with the reference to the associated papers.

Chapter 2
In chapter 2, the background material is given. We describe the mathematical tools that will be useful in understanding the contributions of this thesis. In particular, we describe the information theoretic tools, cut-set bound analysis, and the coding in finite fields.

Part I: Chapter 3
In summary, in this chapter, we study the repair process while we consider the network topology and the transmission costs between nodes. In such a process, we define the sum of the costs of transmitting packets between all pairs of nodes in repair as the repair-cost, and then we investigate the minimum-cost repair. Moreover, we propose an algorithm in which the optimal repair-cost for an arbitrary network is derived. We propose surviving node cooperation (SNC) method, and show that it can reduce the repair-cost. An upper bound for the finite field size of constructing the optimal codes is derived. We study the impact of network topology in repair. This chapter is based on the following papers:


Part II: Chapter 4

In this chapter, we study the repair problem in distributed storage systems where storage nodes are connected through packet erasure channels and some nodes are dedicated to repair (termed as DR storage nodes). We first investigate the minimum required repair-bandwidth in an asymptotic setup, in which the stored file is assumed to have an infinite size. The result shows that the asymptotic repair-bandwidth over packet erasure channels with a fixed erasure probability has a closed-form relation to the repair-bandwidth in lossless networks. Next, we show the benefits of DR storage nodes in reducing the repair bandwidth, and then we derive the necessary minimal storage space of DR storage nodes. Finally, we study the repair in a non-asymptotic setup, where the stored file size is finite. We study the minimum practical-repair-bandwidth, i.e., the repair-bandwidth for achieving a given probability of successful repair. A combinatorial optimization problem is formulated to provide the optimal practical-repair-bandwidth for a given packet erasure probability. We show the gain of our proposed approaches in reducing the repair-bandwidth. This chapter is based on the following papers:

\[ \text{[GXL}^{+}16\text{]} \] M. Gerami, M. Xiao, J. Li, C. Fischione, and Z. Lin, “Repair for distributed storage systems with packet erasure channels and dedicated nodes for repair,” IEEE Transactions on Communications, vol. 64, no. 4, pp. 1367-1383, April 2016.

\[ \text{[GX}^{13}\text{]} \] M. Gerami, and M. Xiao, “Repair for distributed storage systems with erasure channels,” in Proc. IEEE International Conference on Communications (ICC), 2013.


Part III: Chapter 5

In this chapter, we study a distributed storage system where parts of the stored file fragments in storage nodes may be lost. We denote a storage node that lost a part of its fragments as a faulty storage node and a storage node that did not lose any fragment as a correct storage node. In a process, termed as partial repair, a
1.2. Thesis Scope and Contributions

set of storage nodes (among faulty and correct storage nodes) transmit repairing fragments to the faulty storage nodes. We propose two-layer coding for storing files in the system. We study the minimum partial-repair bandwidth, and the codes that achieve the optimal bound. This chapter is based on the following paper:


Part III: Chapter 6

In this chapter, we study security in a distributed storage system where parts of the stored file fragments in storage nodes may be lost. We first investigate the optimal partial repair in which the required bandwidth for recovering the lost fragments is minimum. Next, we assume that an eavesdropper wiretaps a subset of links between storage nodes, and overhears a number of repairing fragments. We then study secure partial-repair in which the eavesdropper obtains no information from the repairing fragments. We propose codes that are optimal in repair-bandwidth and are also optimal in terms of strong or weak security conditions. We also provide optimal secure codes for exact partial-repair in a special case. We show the gain of our proposed codes compared to random network codes in achieving the optimal security bounds. This chapter is based on the following papers:


Part IV: Chapter 7

In this chapter, we study consistency in read operations in distributed storage systems. Given the probability of receiving a version by a storage node and the constraint on the node storage space, we aim to find the optimal encoding of multi versions of a file, such that the probability of obtaining the latest version of a file or a version close to the latest version is maximized. This chapter is based on the following paper:


1.3 Copyright Notice

As detailed in Section 1.2, parts of the material presented in this thesis have been already published by IEEE and Wiley, and some parts are submitted to IEEE. IEEE and Wiley hold the copyright of the corresponding published papers and will hold the copyright of the corresponding submitted papers if they are accepted. Material is reused in this thesis with permission.
This chapter offers preliminaries in coding and information theory, and the mathematic tools which are useful in understanding the subsequent chapters. In particular, we describe the information theoretic tools that help us to derive fundamental bounds in distributed storage systems.

2.1 Entropy and Mutual Information

Entropy of a random variable $X$, denoted by $H(X)$, is a measure of uncertainty of the random variable. For a discrete random variable $X$ with a probability mass function $p(x)$ and $x \in \mathcal{X}$, the entropy $H(X)$ is defined as

$$H(X) = \sum_{x \in \mathcal{X}} -p(x) \log(p(x)).$$  \hfill (2.1)

Similarly, for two discrete random variables $X, Y$ with a joint probability mass function $p(x, y)$ where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ the joint entropy $H(X, Y)$ is defined as

$$H(X, Y) = \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} -p(x, y) \log(p(x, y)).$$  \hfill (2.2)

Also, for two discrete random variables $X, Y$ with a joint probability mass function $p(x, y)$ where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, the conditional entropy $H(Y|X)$ is defined as

$$H(Y|X) = \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} -p(y|x) \log(p(y|x)).$$  \hfill (2.3)

For two discrete random variables $X, Y$, the mutual information $I(X; Y)$ shows the amount of reduction in uncertainty of $Y$ if we know $X$, and is defined as

$$I(X; Y) = H(Y) - H(Y|X).$$  \hfill (2.4)
If $X$ denotes the input random variable of a channel and $Y$ the output random variable, then the maximum value of mutual information is defined as the capacity of the channel and is denoted by $C$. The maximization is taken over the input channel distribution $p(x)$. More formally,

$$C = \max_{p(x)} I(X; Y).$$

(2.5)

2.2 Binary Erasure Channel

A binary erasure channel, depicted in Figure 2.1, is one of the simplest communication channels in information theory to analyze. It has two inputs 0 and 1, and has three possible outputs: 0, 1 and erasure which is denoted by $e$. In other words, a transmitted bit is either received correctly or erased. This channel was first introduced by Peter Elias in 1955 as a toy example [Eli55]. Although the channel may not exist in real, its simplicity makes it very popular in information theory since it can provide intuition about more realistic yet difficult communication channels. In this channel, the transmitted bit is either erased by a probability $p$ or received correctly by probability $1 - p$. The variable $p$ represented the channel erasure probability. We can derive the capacity of this channel by the tools that we discussed recently.

**Theorem 2.1.** The capacity of a binary erasure channel with erasure probability $p$ is $1 - p$ bits per time unit, and is achieved when the channel input has a uniform distribution over input alphabets.
2.3. Packet Erasure Channel

Proof.

\[ C = \max_{p(x)} I(X;Y) \]  
\[ = \max_{p(x)} H(Y) - H(Y|X) \]  
\[ = \max_{p(x)} H(Y) - H(p) \]  
\[ = \max_{Pr(X=0)} (1-p)H(Pr(X=0)) + H(p) - H(p) \]  
\[ = \max_{Pr(X=0)} (1-p)H(\text{Pr}(X=0)) \]  
\[ = (1-p) \]  

where (2.8) holds because \( H(Y|X) = H(p) = -p \log p - (1-p) \log (1-p) \), (2.9) holds because \( H(Y) = (1-p)H(\text{Pr}(X=0)) + H(p) \), and (2.10) holds because \( \max H(\text{Pr}(X=0)) = 1 \) for uniform channel input distribution. When there is a feedback channel from the receiver to the transmitter, the above capacity can be achieved by infinite retransmission [CT12]. The above capacity can also be achieved if there is no feedback channel by encoding the message bits by rateless codes [Mac05].

We use the above results to derive the capacity of packet erasure channels, which is defined in the next subsection.

2.3 Packet Erasure Channel

A fixed number of bits typically constitute a packet. To protect information bits in packets, the bits are generally encoded by a cyclic redundancy check (CRC) codes. Then, a packet is either considered as correctly received packet if CRC can correct error bits or considered as erased packets otherwise. When a packet is erased, we assume all the data in the packet is lost. For a packet erasure channel with packet erasure probability \( p \), we can derive the capacity of the channel, based on the same arguments as the binary erasure channel. Consequently, the capacity of a packet erasure channel with a packet erasure probability \( p \) is \( (1-p) \) packets per time unit. Again, the capacity can be achieved by retransmission if there exists a feedback channel or by rateless codes.

2.4 Network Coding

Consider a network of nodes where a number of source nodes want to transmit information to a number of destination nodes. Network coding means that the intermediate nodes in the network are allowed to transmit a function of their received packets on their outgoing links. Thus, it is a generalized form of the traditional and simple method of store-and-forward. This notion of network coding was introduced
by Ahlswede et al. in [ACLY00]. The result in [ACLY00] is specially interesting for multicast networks, where they showed that the optimal transmission rate cannot be achieved by store-and-forward method and network coding is necessary to achieve the optimal rate. While in [ACLY00] the intermediate node can use any function for encoding its received packets, later Li et al. in [LYC03] showed that linear network coding is sufficient to achieve the optimal multicast rate, namely multicast capacity. Ho et al. in [HMK+06] showed that random linear network coding also achieves the multicast capacity if the code alphabet size is large enough. Koetter et al. in [KM03] proposed an algebraic structure which unifies the previous results in network coding and also extends the previous results.

Network coding can benefit networks in throughput, robustness, scalability and security [HL08]. The gain in throughput is generally shown by the butterfly network introduced in [ACLY00]. The butterfly network has been shown in Figure 2.2. In this figure, a source node denoted by $S$ wants to transmit packets $b_1$ and $b_2$ to the destination nodes $T_1$ and $T_2$ in one time unit. If coding is not allowed in the intermediate nodes, both packets $b_1$ and $b_2$ cannot be obtained at the destination nodes at one time unit. Whereas, if coding is allowed at the intermediate nodes, then node 3 can linearly combine packets $b_1$ and $b_2$ and transmit encoded packet $b_1 + b_2$ to node 4. Then, the destination nodes can decode packets $b_1$ and $b_2$ in one time unit. In general, cut analysis in multicast capacity gives the maximum rate of information from a source node to destinations. This is described in the following definition.
2.5 Secure Network Coding

Definition 2.1 (Cut capacity). Consider a single-source single destination graph $G(V, E)$, where $V$ is the set of nodes and $E$ the set of edges and each edge $e \in E$ has capacity of $c_e$. A cut between a source and a destination refers to a set of edges in which network nodes are divided into two complementary sets of nodes (let us say sets $Q$ and $\overline{Q}$); one set contains the source (let us say set $Q$) and the other set contains the destination node. The value of a cut is the sum of the capacities of the edges from the source to the destination (that is, from $Q$ to $\overline{Q}$). A cut with the minimum value is defined as the min-cut.

In a single-source single-destination network, the store-and-forward approach achieves the optimal rate, and the optimal rate is determined by cut analysis, as stated in the following theorem.

Theorem 2.2 (Max-flow Min-cut Theorem [FF62]). Consider a single-source single destination graph $G(V, E)$, where $V$ is the set of nodes and $E$ the set of edges and each edge $e \in E$ has the capacity of $c_e$. The maximum flow from the source to the destination node is equal to the capacity of the min-cut.

For a single-source multicast network, the optimal rate can be achieved by network coding, and the optimal rate is determined by cut analysis, as stated in the following theorem.

Theorem 2.3 (Max-flow Theorem in Multicast Networks [Yeu08]). Consider a single-source multicast network with a graph $G(V, E)$, where $V$ is the set of nodes and $E$ the set of edges and each edge $e \in E$ has a capacity of $c_e$. The maximum information rate from the source to the destinations is equal to the minimum cut capacity from the source to the destinations.

2.5 Secure Network Coding

In this section, we overview the main results in the literature of network security for which network links are error-free. We firstly study security over the simplest network, i.e., the error-free point-to-point channel, known as the wiretap channel type II. Then, we state the main results for strongly and weakly secure network coding over multicast networks.

2.5.1 Wiretap Channel Type II

The security problem on wiretap channel type II is studied by Ozarow and Wyner in [OW84]. In this problem, there exist a transmitter, a receiver and an eavesdropper (intruder) and error-free channels between transmitter and receiver, as well as between transmitter and eavesdropper. The transmitter encodes $k$ message symbols to $n$ symbols ($n > k$) and then sends $n$ encoded symbols to the receiver over an error-free channel. An eavesdropper overhears $\mu$ symbols of his choice from the $n$ transmitted symbols. This channel is depicted in Figure 2.3. The goal is to design
an encoder such that the receiver can decode the $k$ message symbols by receiving $n$ symbols while the eavesdropper cannot decode any information by overhearing $\mu$ symbols where $\mu < n$. More formally, suppose that $S$ denotes the random variable associated with the $k$ message symbols $(s_1, s_2, \ldots, s_k)$, $X$ denotes the random variable associated with the $n$ encoded symbols $(x_1, x_2, \ldots, x_n)$, and $E$ denotes the random variable associated with the $\mu$ overhead symbols $(x_i_1, x_i_2, \ldots, x_i_\mu)$ by the eavesdropper. Then, for strong security we must have

$$H(S|E) = H(S), \quad (2.12)$$

meaning that by knowing $E$, the uncertainty about the source is not reduced. For perfectly decoding the message symbols at the receiver, we must have,

$$H(S|X) = 0. \quad (2.13)$$

Ozarow and Wyner showed that for $\mu \leq n - k$ we can design an encoder such that conditions (2.12)-(2.13) are satisfied. For that, the encoder uses coset codes. Assume each symbol is taken from $GF(q)$, where $q$ is the finite field size. Coset codes partitions the $q^n$ vector space to $q^k$ partitions. Each partition represents one message vector and the message vector in a partition can be decoded by a parity check matrix $H$ with dimension $k \times n$. The encoder selects a codeword randomly and uniformly from the set of vectors in a partition. For illustration, in Table 2.1 we show the encoded symbols for $k = 1$, $n = 2$, and $\mu = 1$, where the parity check matrix is $H = [1 1]$. In general, a proper parity check matrix can be found by an $(n, n-k)$ MDS code \cite{RSS12}. 

### Table 2.1: A coset code for $n = 2, k = 1, \mu = 1$.  

<table>
<thead>
<tr>
<th>message symbol</th>
<th>transmitted vector 1</th>
<th>transmitted vector 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(10)</td>
<td>(01)</td>
</tr>
<tr>
<td>0</td>
<td>(11)</td>
<td>(00)</td>
</tr>
</tbody>
</table>
2.5. Secure Network Coding

Figure 2.4: Data security in the butterfly network if network coding is used at the intermediate nodes.

2.5.2 Strong Security Over Wiretap Network II

A wiretap network type II is a generalized model of the previously studied wiretap channel. In the wiretap network, there is a source node, and a number of destination nodes in a multicast network with error-free links. Each link has capacity of one unit. The source node encodes $k$ message symbols to $n$ symbols. Each of the destination nodes receives $n$ symbols without any error. An eavesdropper overhears $\mu$ links of its selection. If there was not network coding in intermediate nodes, we could simply apply the coding in wiretap channels and have a secure network. However, network coding may destroy the security as we show in an example in Figure 2.4 adapted from [ERSS12]. In this example, $k = 1$, $n = 2$, and $\mu = 1$. Here, we see that an eavesdropper who overhears the link $2 \to 4$ can decode the source file, if the source uses coset codes with parity check matrix $H = [1 \ 1]$. In this example, if we want to secure the network, we must use global encoding vectors on links such that they are independent of row vectors in $H = [1 \ 1]$.

In general, a multicast network can be secured following the same argument as the above example. This is stated in the following theorem.

**Theorem 2.4** (Theorem 1 in [ERSS12]). Consider an acyclic multicast graph $\mathcal{G}$ with unit capacity edges. Suppose that the source node encodes $k$ symbols into $n$ symbols (where $n \geq k$) and sends the encoded symbols to destination nodes, where the min-cut capacity of the graph is $n$. If the global encoding vectors of $\mu = n - k$ edges are not in the vector space spanned by row vectors of a parity check matrix
H, then the eavesdropper who overhears at most $\mu$ edges cannot decode any information about the source. More formally, if $C_w$ denotes the global encoding vectors of overheard edges by the eavesdropper, for strong security we must have

$$\text{rank} \begin{pmatrix} C_w \\ H \end{pmatrix} = n \quad \text{for all } C_w \text{ s.t. rank } C_w = \mu.$$  \hspace{1cm} (2.14)

### 2.5.3 Weak Security Over Wiretap Network II

Let us first describe the concept of weak security. Consider a source node that has two bits $S = \{b_1, b_2\}$ which are randomly and uniformly selected from $GF(2)$. Let $S$ denote the random variable associated with the source, $S_1$, and $S_2$ denote the random variables associated with the source symbols $b_1$, and $b_2$, and let $E$ denote the random variable associated with the symbol observed by an eavesdropper. If the eavesdropper obtains the encoded symbol $b_1 + b_2$ in $GF(2)$, it obtains one bit of information about the source, i.e., $I(S; E) = 1$ bit. However, the eavesdropper cannot obtain any meaningful information about source symbols $b_1$ and $b_2$ by having access to $b_1 + b_2$. More formally, $I(S_i; E) = 0$, for $i = 1, 2$. This type of security, which was introduced by Bhattad and Narayanan in [BN05], is known as weak security.

A multicast network can be made weakly secure, by a proper precoding as stated in the following theorem.

**Theorem 2.5** (Theorem 1 in [BN05]). Consider an acyclic multicast graph $G$ with unit capacity edges. Suppose that the min-cut capacity of the graph is $n$ and an eavesdropper overhears at most $\mu$ edges of the graph. If $r$ is the maximum rank of global encoding vectors of selecting $\mu$ edges of the graph, then there exists a precoding matrix of dimension $n \times n$ with elements from $GF(q)$ such that the multicast network is weakly secure, if

$$q^n > |A|q^r + q^{n-1}.$$  \hspace{1cm} (2.15)

### 2.6 Coding in Storage Systems

Coding in storage systems has a long history, since the first used codes in storage systems return to the use of Reed-Solomon codes. These codes were introduced by Irving S. Reed and Gustave Solomon in 1960. Reed-Solomon codes still are widely used in CDs, DVDs, and Blu-ray Discs.

There is a property in Reed-Solomon codes which makes them appealing for applications. That is, Reed-Solomon codes optimally use the given redundancy to provide reliability. From this sense, Reed-Solomon codes belong to a more general class of codes termed as maximum distance separable (MDS) codes. These codes have the maximum error correction and detection capabilities. More specifically,
suppose that \( d_{\text{min}} \) is denoted as the minimum distance of a code, which is defined as follows.

**Definition 2.2.** Let \( C \) be a linear \( q \)-ary code with block length \( n \), information length \( k \). Suppose that \( x = (x_1, x_2, \ldots, x_n) \) and \( y = (y_1, y_2, \ldots, y_n) \) are two codewords of the code \( C \). The Hamming distance between these two codewords is

\[
d(x, y) = \{|1 \leq i \leq n | x_i \neq y_i|\},
\]

and the minimum distance of a code \( C \) is defined as

\[
d_{\text{min}} = \min\{ d(x, y) | x \in C, y \in C \text{ and } x \neq y \}.
\]

**Theorem 2.6 (Singleton bound).** Let \( C \) be a linear \( q \)-ary code with block length \( n \), information length \( k \), and a minimum distance \( d_{\text{min}} \), then we have

\[
d_{\text{min}} \leq n - k + 1.
\]

**Proof.** Since the minimum distance of the code is \( d_{\text{min}} \), if we remove \( d - 1 \) symbols from last symbols in the codewords, still there will be distinct codewords. The total number of these codewords are \( q^{n-d+1} \). When we have \( k \) information symbols we must have

\[
q^k \leq q^{n-d_{\text{min}}+1}.
\]

This finalizes the proof.

Codes that satisfy the Singleton bound are denoted as MDS codes and for these codes

\[
d_{\text{min}} = n - k + 1,
\]

meaning that the codewords can be correctly decoded even if they face maximally \( n - k \) erasures.

Reed-Solomon codes can be constructed by Vandermonde or Cauchy matrices. Let \( m = (m_1, m_2, \ldots, m_k) \) be a message vector. The encoding function \( E: q^k \rightarrow q^n \) of a Reed-Solomon code using a Vandermonde matrix is defined as

\[
E(m) = mA,
\]

where \( A \) is \( k \times n \)-dimensional matrix as

\[
A = \begin{pmatrix}
1 & 1 & \ldots & 1 \\
\alpha_1 & \alpha_2 & \ldots & \alpha_n \\
\vdots & \ddots & \vdots \\
\alpha_1^{k-1} & \alpha_2^{k-1} & \ldots & \alpha_n^{k-1}
\end{pmatrix}.
\]
The elements $\alpha_1, \alpha_2, \ldots, \alpha_n$ are distinct elements taken from $GF(q)$. Matrix $A$ is transpose of a Vandermonde matrix.

Although Reed-Solomon codes are very efficient codes in the use of redundancy, they have high computational complexity in encoding and decoding, since a multiplication operation in a finite field is expensive. Due to this fact, MDS array codes have been introduced which basically use XOR operations in the binary field for the encoding and decoding. Several classes of these codes, e.g., EVENODD codes [BBV96, BBM95], B-codes [KBBW99], X-codes [KB99], RDP codes [CEG+04] and Star codes [HX08] have been introduced recently.

Although the aforementioned XOR based codes are simpler in encoding and decoding compared to Reed-Solomon codes, they still have high decoding complexity. By emerging Tornado codes [Lub98, Lub02] by Luby in 1997, research attention has been also on designing near-optimal codes that have very low complexity in decoding [Sho06, Mac05, PY04, DM98]. These codes use message passing decoder which is known for its very low complexity. The disadvantage of these codes is that the decoder must download $k(1 + \epsilon)$ messages for decoding $k$ messages where $\epsilon > 0$. Yet, due to their low encoding/decoding complexity, these codes have been the focus of research for coding in storage systems [WL08, HCL13, DPR06, DPR05, AD14, AGG15, DO11].

2.6.1 Regenerating codes

The codes that we mentioned in the previous section are not efficient in the sense of the required bandwidth in repair. While node failure is a normal catastrophe in distributed storage systems [DGW+10, DRWS11a, SAP+13], it is important to study the costs of repairing a failed node. To illustrate, consider that in a 3000 node production cluster of Facebook, around 20 node failures happen in a day [SAP+13]. In a system that uses a Reed-Solomon code, when a node fails, a new node can be generated by downloading all the file to the new node and then recovering the lost data. This scheme of repairing is not efficient. Consider that if a file of size $M$ bits is encoded by an $(n, k)$-MDS code, then each storage node stores $M/k$ bits and every $k$ storage nodes can reconstruct the original file. Using the above scheme in repair means downloading $M$ bits for generating $M/k$ bits which does not look efficient especially for a large $M$ and $k$.

In the sense of the number of bits transmitted between storage nodes in repair, denoted as repair-bandwidth, replication is optimal. Replication denotes as copying the content of a storage node in several other storage nodes without any source file encoding. Since in repair, the copy which has the same size of the lost data is transmitted to the new node and no other scheme can do better than this. However, as we mentioned replication is not an efficient method in the use of redundancy for providing reliability. Thus, a question arises that: Is it possible to design erasure codes that have smaller repair-bandwidth than transmitting the total file to the new node?

Dimakis et al. answered this question in their pioneering work [DGW+10]. They
formulated the repair problem using information-theory tools and showed that we can have erasure codes that are optimal in repair bandwidth. These codes are denoted as regenerating codes. They also showed that by slightly increasing the stored data in each storage node, the repair bandwidth can be considerably reduced. We state their main results in the following theorem.

**Theorem 2.7** (Bandwidth-storage tradeoff [DGW+10]). Consider a distributed storage system that stores a file of size $M$ bits in $n$ storage nodes such that each storage node stores $\alpha \geq M/k$ bits and every $k$ storage nodes can reconstruct the original file. In repair, $d$ surviving nodes individually transmit $\beta$ bits to the new node. Let $\gamma$ denote the total bandwidth in repair, i.e., $\gamma = d\beta$. If $\alpha \geq \alpha^*(n,k,d,\gamma)$ then there exists a linear code to achieve the bound. Otherwise, it is information-theoretically impossible to achieve points for $\alpha < \alpha^*(n,k,d,\gamma)$. The optimal bound $\alpha^*(n,k,d,\gamma)$ is computed by the following equations.

\[
\alpha^*(n,k,d,\gamma) = \begin{cases} 
\frac{M}{2M - g(i)} & \text{if } \gamma \in [f(0), +\infty) \\
\frac{M - g(i)}{k - i} & \text{if } \gamma \in [f(i), f(i-1)),
\end{cases}
\]  

(2.23)

where

\[
f(i) \triangleq \frac{2Md}{(2k - i - 1)i + 2k(d - k + 1)},
\]  

(2.24)

\[
g(i) \triangleq \frac{(2d - 2k + i + 1)i}{2d},
\]  

(2.25)

for $d \leq n - 1$.

Figure 2.5 shows the fundamental bandwidth-storage tradeoff for $n = 10$, and $k = 5$. The codes achieving two extreme points on the fundamental bandwidth-storage tradeoff are termed as minimum storage regenerating (MSR) and minimum storage.
Background

bandwidth regenerating (MBR) codes. These two points can also be derived by two sequential optimization processes. MSR codes are achieved by first minimizing the storage and then minimizing the repair-bandwidth. The minimum storage capacity required for the reconstruction property is $M/k$. Thus, storage nodes by MSR codes store the same amount of data as the MDS codes. However, MSR codes have the minimum bandwidth in regenerating a new node. We can derive the minimum repair-bandwidth for an MSR code as

$$\alpha_{\text{MSR}} = \frac{M}{k}, \quad \gamma_{\text{MSR}} = \frac{Md}{k(d-k+1)}.$$  \hfill (2.26)

In the optimization process, if we first minimize the repair-bandwidth and then storage per each node, another extreme point, the MBR point, is achieved. In general $\gamma \geq \alpha$ [SRKR12]. For MBR codes $\gamma = \alpha$. Therefore, setting $\gamma = d\beta = \alpha$ on the optimum bound $\sum_{i=0}^{k-1} \min\{\alpha, (d-i)\beta\} = M$ yields

$$\alpha_{\text{MBR}} = \frac{2Md}{k(2d-k+1)}, \quad \gamma_{\text{MBR}} = \frac{2Md}{k(2d-k+1)}.$$  \hfill (2.27)

2.7 Consistency in Distributed Storage Systems

In the existing large-scale distributed storage systems, e.g., in Amazon [DHJ+07] and Cassandra [Cas13], the stored data is typically replicated in several storage nodes. Replication provides reliability to the systems since if a storage node fails or a part of the system is not available, then there is still a chance of accessing the file by the replicated nodes. Replication also improves performance of distributed storage systems since it provides the data closer to users and then the user face smaller latency in accessing a file. Replication is also beneficial for scalability since different user requests can be responded by distinct servers.

On the other hand, replication brings new challenges. One of the main challenges is on consistency of data in distributed storage systems. When data in one node changes, the change should be reported to other replicated nodes and then their data should be updated. If the updates are not performed in all replicas, then users that connect to different nodes may observe different data, thus creating inconsistency in the systems. In large-scale distributed storage systems, the storage nodes are distributed around the world. Then the process of updating all replicas takes long time. In this condition, one can choose one of the following approaches.

First approach is to freeze all other read and write operations in the system and complete a change by updating all replicas. By this approach, all the read operations return the latest version of data and the system is strictly consistent, however the system might not be available in the period of updating replicas. In other words, we sacrifice availability for the consistency. This in e-commerce business implies significant financial loss and also affects customer trust [DHJ+07].
the systems that choose consistency against availability are databases that have transactions fulfilling ACID properties. ACID in the computer-science literature stands for atomicity, consistency, isolation, and durability. Atomicity means that a transaction is either completed or failed. Consistency means that the system goes from one valid state to another valid state. Isolation means that concurrent transactions are run in such a way that is the same as transactions performed serially. Durability means that a committed transaction will stay in the system forever, even if there is failure in part of the network.

Second solution is to loosen consistency, then users may observe different versions. Later, in the background updating replicas is performed. In this way, consistency is sacrificed for availability. Among the systems that choose availability against consistency are databases that have transactions fulfilling BASE (basic availability, soft-state, and eventual consistency) properties.

In the computer-science literature there is a known tradeoff between consistency and availability. It is stated in a theorem, known as CAP (consistency, availability and partition tolerance) theorem, that for a system that is partition tolerant, one cannot have both of consistency and availability.

2.7.1 Coding in Consistent Distributed Storage Systems

The effect of coding in distributed storage systems where the source information changes has been studied recently [ASV10, MWC12, ERGKM15, HDO15]. In a system that uses erasure codes for storing data, it is more efficient to use codes that have small changes by slightly changing the information packets. Then, communication cost is reduced in the process of update. These codes are termed as update efficient codes [ASV10]. In another line of research, multi-version codes are studied in distributed storage systems where different versions of a file are assumed to be independent and different storage nodes may receive a subset of different versions [WC14b, WC14a, Kha15b].
Part I

Surviving Nodes Cooperation in Repair
Chapter 3

Optimal-cost Repair with Surviving Nodes Cooperation

This chapter studies the transmission cost of repair in a distributed storage system, where storage nodes are connected together through an arbitrary network topology, and there is a cost in the use of the network link. Contrary to the classical model, where there exists a link between a pair of storage node, in our repair model there might not exist a link between some pairs of storage nodes or it might be expensive to use. For that, we propose surviving nodes cooperation in repair, meaning that the surviving nodes as the intermediate nodes combine their received packets with their own stored packets, and then transmit coded packets toward the new node. We show that surviving nodes cooperation can reduce the repair-cost, the sum of the costs for transmitting repairing data between the surviving nodes and the new node. For the system that allows surviving node cooperation, we find the minimum-cost codes in repair by firstly deriving a lower bound of the repair-cost through an optimization problem and then proposing achievable codes. We show the gain of the proposed codes in reducing the repair-cost in some scenarios.

3.1 Introduction

The study on the role of coding in distributed storage systems has recently attracted lots of research interest. Due to the vast applications of these systems, e.g., in data-centers, peer-to-peer networks and wireless sensor networks (WSNs), it will be interesting to investigate the advantages and the costs of coding in these systems. In the recent work, in [WK02], the advantage of erasure codes in providing higher reliability compared to replication has been presented, while both schemes use the same storage space. In [PlaMREG+03, DLS+04, SAP+13, BTC+04, SDKB14, Haf05, HX08, XB99, BBBM95] erasure codes, based on Reed-Solomon codes, parity array codes and LDPC codes, have been designed for some applications, e.g., for HDFS RAID systems in Facebook [SAP+13]. In [MPBA10, MPA+11] random network coding is proposed for generating redundancy in Wuala peer-to-peer networks. Among
the codes, maximum distance separable (MDS) codes offer the highest reliability in the use redundancy.

However, the above advantage of coding does not come free and coding compared to replication may impose higher costs to the storage systems in some scenarios, e.g., in repair. In distributed storage systems, when a node fails, to maintain the reliability of the system, a new node is generated. The process of generating the new node is termed as the repair process. Recently, the costs in repair has been studied from different aspects, e.g., in repair bandwidth [DGW+10], the number of disk I/O reads [ERR10], and repair locality [PD14]. Dimakis et al. in [DGW+10] studied the required number of bits in repair, denoted as repair bandwidth, and derived the minimum repair bandwidth. A new class of erasure codes, namely regenerating codes based on network coding [ACLY00, KM03], are proposed in [DGW+10, Wu10a]. In the proposed codes, the new node may not have the same encoded data as the failed node; however, the new node and the surviving nodes still preserve the property that every $k$ nodes can reconstruct the original file. This kind of repair is termed as functional repair. The exact regenerating of a new node has been studied in [RSKR09a, SRKR12]. Since the proposed approach in [DGW+10] uses network coding to achieve the optimal repair-bandwidth codes, it requires multiple reads from any storage node. To reduce the number of reads, fractional repetition codes has been introduced by El Rouayheb and Ramchandran in [ERR10]. Another important criterion is the number of surviving nodes that are connected in repair, denoted as the repair locality. The repair locality has been studied by Papailiopoulos and Dimakis in [PD14]. Finally, another important criterion in repair is the transmission cost. We study the transmission cost in the repair in a distributed storage system whose nodes are connected through an arbitrary network topology and network links have different costs.

References [DGW+10, Wu10a, RSKR09a, HXW+10, ERR10, PD14] have well addressed the optimal repair from different aspects. However, the link cost (transmission cost in channels) and the impact of the network topology have not been considered. In a practical system, the transmission cost is an important design consideration and different links (channels) of a network may have different costs. For instance, in data-centers data transmission between different pairs of storage nodes may have different costs due to the different paths data passes through switches/routers [RBP+11, AFLV08]. The topology of a network may also impact the repair cost. Recently, reference [AKG10] considers the transmission cost from the surviving nodes to the new node and the cost-bandwidth tradeoff is derived. Yet, this model does not exploit the network topology in repair. In recent works, Martalo et al. in [MPBA10, MPA+11] studied random network coding in overlay architecture of Wuala networks and showed the benefits of random network coding in generating the new fragments. Yet, the authors in [MPBA10, MPA+11] did not

---

*1A file of size $M$ bits when coded by an $(n, k)$-MDS code, it is divided to $k$ parts and then encoded to $n$ parts, such that any part has $M/k$ bits and any subset of $k$ parts can reconstruct the original file.*
formulate an optimization problem over a general network topology to minimize
the repair-cost. We address the impact of network topology and transmission cost
in repair. In summary, the main contributions of this chapter are:

- We study the repair process while we consider the network topology and the
  transmission costs between nodes. In such a system, we define the sum of
  the costs of transmitting packets between all pairs of nodes in repair as the
  repair-cost, and then we investigate the minimum-cost repair.

- We propose an algorithm in which the optimal repair-cost for an arbitrary
  network is derived.

- We propose surviving nodes cooperation (SNC) method in repair, and show
  that it can reduce the repair-cost.

- An upper bound of the finite field size for constructing the optimal codes is
  derived.

- We study the impact of network topology in repair.

The remainder of this chapter is organized as follows. In Section 3.3, we formu-
late an optimization problem that offers a lower bound of the repair-cost. Further,
in Section 3.4, we show that the lower bound is achievable, and we characterize
the sufficient finite field size for the codes that achieve the bound. In Section 3.5, we
apply the proposed algorithm in large scale distributed storage systems. Finally, in
Section 4.7, we conclude the chapter.

3.2 Motivating Examples

Before we formally state the problem, we give two motivating examples. Firstly,
consider a distributed storage system in Figure 3.1. The system stores a file con-
taining two fragments \( \{a, b\} \). The stored fragments are encoded by a \((3, 2)\)-MDS
code. That is, every two storage nodes can reconstruct the original file. In this
example, node 3 fails and the new node is generated by the help of nodes 1 and
2. Assume, transmitting a fragment from nodes 1 and 2 to the new node costs 10
units and transmitting a fragment from node 1 to node 2 costs 1 units. In practice,
these various costs can come from the fact that node 1 and node 2 are in the same
storage rack, while the new node located in another rack. In classical repair model,
node 1 and 2 send fragments \( a \) and \( b \) to the new node to generate the fragment \( a + b \)
in the new node. This costs \( 10 + 10 = 20 \) units. When surviving node can cooperate
in repair, node 1 sends the fragment \( a \) to node 2. Node 2 then combines its received
fragment with its stored fragment and generate the fragment \( a + b \), and sent it to
the new node. The repair with the surviving node cooperation costs \( 1 + 10 = 11 \)
units. This shows that surviving node cooperation can reduce the repair-cost from
20 units to 11 units, in this example.
Secondly, we study a distributed storage system over a tandem network. For that, let us define a tandem storage network.

**Definition 3.1 (Tandem storage network).** In a tandem storage network, storage nodes are arranged in a line. All the storage nodes have two neighboring nodes except two nodes at the end which have only one neighbor.

Consider a distributed storage system in a four-node tandem network shown in Figure 3.2. A source file is divided into four equal size fragments. Assume a file of size 4 mega-bits (or equivalently $M = 4$ fragments if one fragment has 1 mega-bits) is coded with an MDS code and distributed among 4 nodes ($n = 4$) such that each node stores 2 one-mega-bit fragments ($\alpha = 2$) and the source file can be reconstructed by any 2 nodes ($k = 2$). When a node fails (say node 4), a new node downloads $\beta$ fragments from each of 3 surviving nodes ($d = 3$).

Let us first model the repair by assuming that there exist direct links between surviving nodes and the new node (classical repair model). The distributed storage system is represented by a directed acyclic graph, known as information flow graph, as shown in Figure 3.3 for the failure/repair on node 4. In the graph, there is a source node (denoted as $S$) connected to storage nodes through infinite-capacity links. Each storage node is depicted by input ($\text{in}$) and output ($\text{out}$) nodes connected by an $\alpha$-capacity link. When a node fails, a new node downloads $\beta$ fragments from each of $d$ surviving nodes. A data collector (denoted as DC) can reconstruct the source file by contacting any $k$ storage nodes.

Our main tool is the cut-set bound analysis on the information flow graph. For DC to have access to the source file, the value of cuts (and accordingly the value of the min-cut) must be greater than, or equal to, the size of the source file. This constraint is called a cut-set bound constraint (or briefly a cut constraint). Satisfying all the cut constraints for a multicast network ensures the existence of a linear code to recover the source file [ACLY00].

By the cut-set bound analysis of the information flow graph in Figure 3.3 we can see that for DC to reconstruct the source file, it requires $\alpha + 2\beta \geq M$. For
3.2. Motivating Examples

Figure 3.2: A distributed storage system in a 4-node tandem network. Dashed lines denote how network codewords are formed. Node 4 fails and node 5 is the new node. For regenerating a new node, $p_1, p_2, p_3$ are formed by linear combination of fragments in node 1, 2, and 3, respectively. The underlying finite field is $\mathbb{F}_{11}$.

$\begin{align*}
S & \xrightarrow{\times 1} p_1 = a_1 + 2b_1 \\
\node1 & \xrightarrow{\times 2} p_2 = 2a_2 + b_2 \\
\node2 & \xrightarrow{\times 3} p_3 = 4a_1 + 5b_1 + 4a_2 + 5b_2 \\
\node3 & \xrightarrow{\times 4} p_4 = 6a_1 + 7b_1 + 8a_2 + 7b_2 \\
\node4 & \xrightarrow{\times 5} p_5 = 6a_1 + 8b_1 + 6a_2 + 6b_2
\end{align*}$

Figure 3.3: Information flow graph if the classical repair model is deployed. Node 4 fails and node 5 is the new node. A cut-set bound analysis in the information flow graph states that there must be $\alpha + 2\beta \geq M$ for successful recovery.
Optimal-cost Repair with Surviving Nodes Cooperation

Figure 3.4: Regenerating by surviving node cooperation in a tandem network. For regenerating a new node, fragment $p_1$ is formed by combining fragments of node 1. Fragments $p_2, p_3$ are formed by linear combination of fragment $p_1$ and stored fragments in node 2. Finally, fragments $p_4, p_5$ are formed by linear combination of $p_2, p_3$, received fragments on node 3, with the stored fragments on node 3.

$\alpha = 2, M = 4$, then $\beta \geq 1$. That is, the new node must download at least $\beta = 1$ mega-bits from each surviving node. The optimal repair-bandwidth to achieve this lower bound is to download $p_1, p_2, p_3$ as in Figure 3.1 from node 1, node 2 and node 3, respectively. Here $p_1, p_2$ and $p_3$ are formed by linear coding at nodes 1, 2, and 3, respectively.

Now, let us study the cost of repair if the classical repair model is deployed. Assume that transmitting one fragment from node 1 to 2, 2 to 3, 3 to 5 costs respectively $C_{12}, C_{23},$ and $C_{35}$, all of which are non-negative real numbers. In the network, $p_1$ passes the route (node 1 $\rightarrow$ node 2 $\rightarrow$ node 3 $\rightarrow$ node 5) with the cost of $c_{12} + c_{23} + c_{35}$ units to reach the new node (node 5). Similarly, $p_2$ passes node 2 $\rightarrow$ node 3 $\rightarrow$ node 5 with the cost of $c_{23} + c_{35}$ units, and $p_3$ passes node 3 $\rightarrow$ node 5 with the cost of $c_{35}$ units. Thus, the total cost in the repair is $c_{12} + 2c_{23} + 3c_{35}$ units. Later, we show that this repair-cost can be reduced.

Consider the repair scheme in Figure 3.4 where the regenerating process allows surviving node cooperation (SNC). Here the cooperation means that surviving nodes are allowed to linearly combine their own fragments with the received fragments from other nodes. For instance, at node 2, $p_2$ and $p_3$ are encoded with the fragments of node 3. Then $p_4$ and $p_5$ are stored in the new node. Note that here we only consider functional repair,
3.3 Problem Formulation

in which the regenerated node may not be identical to the failed node but it has the same MDS code property. That is, with the new node (node 5), any 2 out of 4 nodes can reconstruct the source. It is easy to see that the cost of repair is reduced to $c_{12} + 2c_{23} + 2c_{35}$ units as shown in Figure 3.4 (only two fragments are transmitted from node 3 to node 5). The example shows that allowing the surviving node cooperation in the multi-hop network can reduce the repair-cost. We note that the repair in Figure 3.4 still is not optimal in term of the repair-cost. We shall later show the optimal repair-cost in this example.

3.3 Problem Formulation

Above, for two specific networks, we have studied the repair-cost and proposed coding at the intermediate nodes to reduce the cost. A natural question is that what the optimal repair-cost is and how to design codes that achieve the optimal point for more general scenarios. In this section, we formulate an optimization problem which establishes a fundamental lower bound on the repair-cost for an arbitrary network.

3.3.1 System Model

Consider a storage system with the original file of size $M$ (measured in fragments) distributed among $n$ nodes in which each node stores $\alpha$ fragments and any $k$ of $n$ nodes can rebuild the original file. We focus on the distributed storage system that stores the file by an MDS code. We denote the source file by an $M \times 1$ vector, $s$. Vector $s$ consists of elements from a finite field $\mathbb{F}_q$, where $q$ is the finite field size. Then, the code on each node $i$ can be represented by a matrix $Q_i = (q_{i1}^1, \ldots, q_{i\alpha}^\alpha) \in \mathbb{F}_q^{M \times \alpha}$, where each column $(q_j^i)$ represents the code coefficients of fragment $j$ on node $i$. Thus, the coded data in node $i$ is $x_i = Q_i^T s$.

Next, we model the information flow in a repair process by a directed acyclic graph $G(\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ is the set of directed links. The information flow graph consists of three different nodes: a source node, storage nodes, and several data collectors (DCs). The source node distributes the original file among storage nodes by the (presumably) infinite-capacity links. Every storage node can be denoted by input (in) and output (out) nodes connecting by a link of capacity $\alpha$. Finally, DC reconstructs the original file by connecting to at least $k$ storage nodes via infinite-capacity links and then recover the stored file from $k$ linear equations by, e.g., Gaussian elimination method. In contrast to the classical repair model [DGW+10], there might not exist links between some of the surviving nodes and the new node, and information from surviving nodes passes a number of intermediate nodes to reach to the new node. Meanwhile, the intermediate nodes are allowed to transmit codewords, which are functions of their received fragments and their stored fragments. We assume intermediate nodes are capable of performing linear operations in the finite fields. The number of surviving nodes in repair, which
is denoted as \( d \), is assumed to be greater than \( k \), i.e., \( d \geq k \) (note that repair for \( d < k \) is information-theoretically impossible). In our model, there is a cost associated with edge \( (ij) \), for \( (ij) \in \mathcal{A} \). The cost of link \( (ij) \), which is denoted as \( c_{(ij)} \), represents the link cost for transmitting one fragment from node \( i \) to node \( j \). We note that \( c_{(ij)} \) is a finite and non-negative real number. We assume all the fragments have equal size and then only consider linear costs\(^2\). This means if the transmission cost of one fragment from node \( i \) to \( j \) is \( c_{(ij)} \), then it costs \( mc_{(ij)} \) to transmit \( m \) fragments from node \( i \) to \( j \). An information flow graph for (one stage of) repair on node 4 in the four-node tandem storage network has been shown in Figure 3.5.

### 3.3.2 Repair-cost Formulation

To study the repair-cost, we define vector \( \mathbf{z} = [z_{(ij)} : (ij) \in \mathcal{A}]^T \), where each element \( z_{(ij)} \) is a non-negative integer number denoting the number of fragments sent from node \( i \) to \( j \) in the repair. Accordingly, we define vector \( \mathbf{c} = [c_{(ij)} : (ij) \in \mathcal{A}]^T \). Hence, the repair-cost, denoted by \( \Gamma \), is computed by

\[
\Gamma = \mathbf{c}^T \mathbf{z}.
\]  

Our objective is to minimize the repair-cost (\( \Gamma \)) in which the system with the new node retains the MDS property. To examine the MDS property of the system after each stage of repair, DC contacts the new node and a set of \( k - 1 \) surviving nodes

\(^2\)When fragments from different files have different sizes, one can define \( c_{ij} \) as the unit cost, and measure the fragment as integral multiple of unit cost.
among \( n - 1 \) storage nodes. This yields \( \binom{n-1}{k-1} \) constraints on one stage of repair. We note that, potentially there are infinite stages of repair. That means after generating a new node, again a node may fail. There, the previously regenerated node (if still alive) can help a currently new node to be generated. Consequently, the network is evolving after each stage of repair. Depending on the network topology and the link costs, the network can evolve for infinite cases. This can make the analysis for a general network complicated. We instead derive a lower bound of repair by the cut-set bound analysis only on the first stage of repair. Then, in the next section, we prove that this bound is achievable for MDS codes.

Thus, we minimize the repair-cost (denoted as \( \Gamma \)) under the constraints that all the cuts of connecting DC to the new node and \( k - 1 \) other storage nodes must be greater than or equal to \( M \), the original file size. For instance, in Figure 3.5, the heavy dotted line illustrates a cut when DC connects to the new node and node 1 for an MDS code with \( k = 2 \). The cut constraint relating to this cut can be formulated by the inequality: \( z_{(35)} + \alpha \geq M \). By assuming vector \( z = [z_{(12)} z_{(23)} z_{(35)}]^T \), we can express the inequality in a vector space as, \((0, 0, 1) z \geq M - \alpha\). Denote \( r \) as the total number of cut constraints when DCs connect to the new node and a set of \( k - 1 \) out of \( n - 1 \) surviving nodes. For each set of \( k - 1 \) surviving selection, if the min-cut on the network is greater than \( M \), then DC have access to the file. Then, \( r = \binom{n-1}{k-1} \).

Denoting \( |A| \) as the cardinality of existing edges between nodes, we form all the inequalities in a matrix form, by defining an \( r \times |A| \) dimensional matrix \( L \) (this matrix is called coefficient matrix \([BV09]\)). The corresponding inequalities induced by the cut constraints show a region in a multi-dimensional space that the subgraph must satisfy to be a feasible solution. This region is often called polytope \([BV09]\). Consequently the polytope is

\[
\Psi = \{ z = [z_{(ij)}] | z_{(ij)} \geq 0, L z \geq b \},
\]

(3.2)

where the comparison of two vectors, e.g., \( a \geq b \) means every element in \( a \) is greater than or equals to the element in \( b \) at the same position.

**Example 3.1.** In Figure 3.5, if DC connects to node 1 and the new node to rebuild the source \( (k = 2) \), the first cut constraint is

\[
z_{(35)} \geq M - \alpha.
\]

(3.3)

The second constraint follows if we connect DC with node 2 and the new node. Hence,

\[
z_{(12)} + z_{(35)} \geq M - \alpha.
\]

(3.4)

Finally, when DC connects to node 3 and the new node, we have the third constraint:

\[
z_{(23)} \geq M - \alpha.
\]

(3.5)
Thus, we can form all these inequalities in a matrix form as

\[
\begin{pmatrix}
0 & 0 & 1 \\
1 & 0 & 1 \\
0 & 1 & 0
\end{pmatrix} \begin{pmatrix}
z_{(12)} \\
z_{(23)} \\
z_{(35)}
\end{pmatrix} \geq \begin{pmatrix}
M - \alpha \\
M - \alpha \\
M - \alpha
\end{pmatrix}.
\]

(3.6)

Since the constraint region and the objective function in the repair problem are linear, and \(z_{(ij)}\)'s are integer numbers then the problem is an integer linear programming problem.

\[
\text{minimize } \Gamma = c^T z \\
\text{subject to } L z \geq b, \quad z_{(ij)} \in \mathbb{Z}^+, \quad (3.7)
\]

where \(\mathbb{Z}^+\) is the set of non-negative integers. An integer linear programming problem has in general high complexity. We apply a relaxation technique by assuming that \(z_{(ij)}\)'s are real numbers. Later, we argue that the loss of optimality by following the above assumption is small if the file size is large. This relaxation transforms the integer linear programming problem into a linear programming problem, as

\[
\text{minimize } \Gamma = c^T z \\
\text{subject to } L z \geq b, \quad z_{(ij)} \in \mathbb{R}^+, \quad (3.8)
\]

where \(\mathbb{R}^+\) is the set of non-negative real numbers. This problem can be solved efficiently, since it has a polynomial-time complexity in number of nodes. Due to the applied relaxation, the above linear programming problem gives a lower bound of the repair-cost.

**Example 3.2.** Consider a four-node distributed storage system in the tandem network, as shown in Figure 3.6. We assume three nodes are cooperating in the repair process \((d = 3)\), \(M = 4, k = 2, \alpha = 2, z = [z_{(12)} z_{(23)} z_{(35)}]\) and the corresponding cost vector is

\[
c = \begin{pmatrix}
c_{(12)} \\
c_{(23)} \\
c_{(35)}
\end{pmatrix}.
\]

(3.9)

In other words, there exists a link between nodes 1 and 2 having cost \(c_{(12)}\) units for transmitting one fragment, and costs \(c_{(23)}\) and \(c_{(35)}\) units for transmitting one fragment respectively from nodes 2 to 3, and from node 3 to 5 (new node). Now we analyze the constraint region for \(M = 4, \alpha = 2\) in (3.7). Hence, we can formulate
the problem as

\[
\begin{align*}
\text{minimize} & \quad \Gamma(z) = c_{12}z_{12} + c_{23}z_{23} + c_{35}z_{35} \\
\text{subject to} & \quad \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} z_{12} \\ z_{23} \\ z_{35} \end{bmatrix} \geq \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}.
\end{align*}
\]

Solving the linear optimization problem (e.g., by the simplex method [BV09]) gives the optimal subgraph \((z_{12}, z_{23}, z_{35}) = (0, 2, 2)\) with the cost of \(2c_{23} + 2c_{35}\) units.

A linear network coding by selecting coefficients (in this example) from \(\mathbb{F}_5\) with SNC in Figure 3.6 can meet the minimum-cost subgraph. The coding scheme is by transmitting fragments \(p_2 = 2a_2 + b_2\), \(p_3 = a_2 + 2b_2\) from node 2 to node 3, and then transmitting fragments \(p_4 = p_2 + (a_1 + b_1 + a_2 + b_2) = a_1 + b_1 + 3a_2 + 2b_2\) and \(p_5 = p_3 + (a_1 + 2b_1 + a_2 + 2b_2) = a_1 + 2b_1 + 2a_2 + 4b_2\) to the new node. Here \(p_4, p_5\) are fragments for the new node. The new node along with the surviving nodes satisfy the MDS property (every \(k\) nodes can reconstruct the original file).

In the previous example, the linear optimization gave the exact optimal solution. That was due to the appropriate selection of the file size, \(M\), such that the linear program outputs integer values for \(z_{(ij)}\)'s. In general we can state the accuracy of the our proposed method in the following proposition.

**Proposition 3.1.** The linear programming problem in (3.8) yields the (exact) optimal-cost repair for large \(M\).
Proof. It is known that if the linear programming problem has a finite optimal solution then the \(z_{ij}\)'s are rational numbers. Since for the optimal-cost problem there is a finite solution (e.g., transmitting all the file to the new node is a feasible solution with a finite cost) then \(z_{ij}\)'s are rational numbers. If the \(z_{ij}\)'s of the optimal solution is rational, then we can make the resulted \(z_{ij}\)'s integer by a proper scaling of the file size. For instance, if for \(M = 10\) (measured in fragments), the optimization problem gives \(z_{ij} = 0.1\), then we scale \(M\) to \(M = 100\). Then, we would have \(z_{ij} = 1\). To be able to scale \(M\), the file size, measured in bits, should be large enough. This is almost always the case for the files stored in large scale data-centers.

Remark 3.1. We note that in the optimal repair-cost problem, \(d\) number of surviving nodes are willing to help regenerating the new node; however, the optimal repair-cost policy may not use all the surviving nodes in the repair, for the matter of costs. For instance, in the above example, we have \(d = 3\), but the cooperation of only two nodes (nodes 2 and 3) gives the optimal repair-cost.

Remark 3.2. We can relate the repair-cost to the consumed energy or bandwidth in repair. Assume that transmitting a fragment from node \(i\) to node \(j\) requires \(c_{ij}\) units of energy. Then transmitting \(z_{ij}\) fragments from node \(i\) to node \(j\) requires \(c_{ij}z_{ij}\) units of energy. By this formulation, \(\sum c_{ij}z_{ij}\) is the total energy consumption in repair. Then, the linear programming problem minimizes the energy consumption in repair. We can also relate the repair-cost to bandwidth. When \(z_{ij}\) is the number of bits transmitted from node \(i\) to node \(j\), then \(\sum z_{ij}\) denotes the total bandwidth in repair. Then, the linear programming problem minimizes the required bandwidth in repair.

3.3.3 Complexity of the proposed algorithm

The proposed algorithm has polynomial-time complexity for a fixed value of \(k\). As previously discussed, the optimal repair-cost subgraph for the proposed algorithm in this chapter is derived by solving a linear programming problem. The formulated linear problem has \(\binom{n-1}{k-1}\) cut constraints, for given \(n\) and \(k\), respectively as the number of storage nodes in the network, and the MDS code parameter. Each cut constraint is resulted from the cut-set bound analysis when DC connects to the new node and a set of \(k - 1\) other storage nodes among \(n - 1\) surviving nodes, which turns out to \(\binom{n-1}{k-1}\) cut constraints. For each selection of \((k - 1)\) storage nodes, we can find the min-cut of the network in polynomial-time, e.g., by the Edmonds-Karp algorithm, which has \(O(|N||A|^2)\) complexity [CLR+01], where \(|N|\) denotes number of nodes in the network and \(|A|\) denotes number of edges in the network.

\(^{3}\)Notation \(f(n) = O(g(n))\) means there exist a constant \(c\) and integer \(I\) such that \(f(n) \leq cg(n)\) for \(n > I\).
Normally $|N|$ and $|A|$ are in polynomial order of $n$. On the other hand, we have
\[
\binom{n-1}{k-1} = \frac{(n-1)(n-2)\cdots(n-k+1)}{(k-1)!} = O(n^{k-1}).
\]

Hence, the complexity of the cut-set analysis for $k = O(1)$, $|N| = O(n)$ and $|A| = O(n^2)$ is polynomial in the number of nodes. Thus, we can find the cut constraints in polynomial time and then solve the resulted linear programming problem in polynomial time.

### 3.4 Achievable Codes for Minimum Repair-cost

In what follows, we shall show that the lower bound of the repair-cost is achievable by a linear code and for $\alpha = M/k$. That is, there exists a linear code corresponding to the repair with the minimum-cost subgraph from the optimization problem (3.8).

Our proof is based on random linear codes and then we discuss the required finite field size for constructing the minimum repair-cost MDS codes. To find the sufficient field size for successful regeneration, we apply sparse-zero lemma as follows.

**Lemma 3.1.** Consider a multi-variable polynomial $g(\alpha_1, \alpha_2, \ldots, \alpha_n)$ which is not identically zero, and has the maximum degree in each variable at most $d_0$. Then, there exist variables $\gamma_1, \gamma_2, \ldots, \gamma_n$ in the finite field $\mathbb{F}_q$, and $q \geq d_0$, such that $g(\gamma_1, \gamma_2, \ldots, \gamma_n) \neq 0$

**Proof.** See proof of Lemma 19.17 in [Yeung].

Suppose a source information file, containing $M$ fragments, is coded by an $(n, k)$-MDS code, that is, each node stores $\alpha = M/k$, and every $k$ nodes can reconstruct the original file. If $x_i$ denotes the stored symbols of node $i$, then $x_i = Q_i^T s$, where $Q_i$ is an $M \times \alpha$-dimensional matrix, which represents the coding coefficients of node $i$. When a node fails (without loss of generality, we assume node 1 fails), solving the optimization problem provides the minimum-cost subgraph. Following the minimum-cost subgraph, the new node is regenerated by the surviving node cooperation. Clearly, with the minimum-cost subgraph, we also know which nodes should encode on the directed information flow graph. Then, by using network codes from a proper finite field, the new node is regenerated. Assume that the coding coefficient matrix of the new node is $Q_1$ and the content of the new node is $x'_1$, then $x'_1 = Q_1^T s$.

To maintain the MDS property after the repair, the coding coefficient matrix $(Q_1)$ have to meet certain requirement. That is, for any selection of $k - 1$ out of $n - 1$ surviving nodes, $\Xi_{k-1} = \{Q_{s_1}, \ldots, Q_{s_{k-1}}\}$, along with the codes of the new node $Q_1$, the polynomial $\det([Q_1, Q_{s_1}, \ldots, Q_{s_{k-1}}])$ is a non-zero polynomial. In what follows, we first show that $\det([Q_1, Q_{s_1}, \ldots, Q_{s_{k-1}}])$ satisfying the subgraph of the optimization process is not identically zero and then discuss the required field size.
Lemma 3.2. For regenerating node 1, there exist linear codes satisfying the minimum-cost subgraph (resulted from problem (3.8)) such that the polynomial $\det([Q_1, Q_{s_1}, \cdots, Q_{s_{k-1}}])$ is non-zero for any selected set $\Xi_{k-1}$. That is,
\[
\prod_{\{s_1, \cdots, s_{k-1}\} \subset \{2, \cdots, n\}} \det([Q_1, Q_{s_1}, \cdots, Q_{s_{k-1}}]) \neq 0. \tag{3.12}
\]

Proof. Consider $\Xi_k = \{Q_{s_1}, \cdots, Q_{s_k}\}$ as a set of coding coefficients selected from $k$ out of $n$ nodes. Since every $k$ nodes can reconstruct the original file, then the matrix $[Q_{s_1}, \cdots, Q_{s_k}]$ has full rank $M = ka$. Thus, for $\Xi_{k-1}$, the matrix $[Q_{s_1}, \cdots, Q_{s_{k-1}}]$ has rank $(k-1)a$. Consider a set $V$ containing the data collector, in and out nodes of the new node, and out nodes of the nodes in set $\Xi_{k-1}$. Other nodes including the source node are in the complement set $V^c$. Since all the cuts has the capacity of $M$ fragments, there would be $R = M - (k-1)a = \alpha$ fragments as the capacity of edges from out nodes of the set $N - \Xi_{k-1}$ to the in node of the new node. In the set $N - \Xi_{k-1}$, there exist $\alpha$ fragments that have independent vectors from the vectors in $\Xi_{k-1}$. If we send those $\alpha$ fragments (through the links with the total capacity $R = \alpha$) to the new node, then the matrix of the coding coefficients will be full rank. Therefore, $\det([Q_1, Q_{s_1}, \cdots, Q_{s_{k-1}}])$ can be non-zero. \hfill \Box

To find the required field size of the codes, we need to know the maximum degree of the variables of the polynomial in (3.12). For analysis, we use $n_{nc}$ to denote the maximum number of encoding processes operated over a fragment in the repair. We note that $n_{nc} \leq |N|$, the number of nodes in the networks.

Example 1: In the repair process of node 4 in tandem network (Figure 3.5), $n_{nc} = 4$. The encoding can be at node 1, 2, 3 and at the new node.

Theorem 3.1. For a distributed storage system $DSS(n, k, \alpha)$ with the source file of size $M$, if the field size is greater than $d_0$, there exists a linear code such that the MDS property is satisfied for any stage of repair, where
\[
d_0 = \binom{n}{k}Mn_{nc}. \tag{3.13}
\]

Proof. The proof is by induction on the number of repair stages. That is, we assume before a node fails all the storage nodes have the MDS property. In each stage of repair, when a node fails, the new node is regenerated preserving the MDS property. Thus, we initialize the code on $n$ nodes by which any $k$ out of $n$ nodes can reconstruct the original file. Then if a node fails, the new node is regenerated such that the repairing cost is minimized and the MDS property is preserved. By the MDS property, the coding coefficients of any $k$ nodes must have full rank $M$. That is,
\[
\prod_{\{s_1, \cdots, s_k\} \subset \{1, \cdots, n\}} \det([Q_{s_1}, \cdots, Q_{s_k}]) \neq 0. \tag{3.14}
\]
3.5 Numerical Results

Figure 3.7: A large scale tandem storage network. The optimal-cost repair is by cooperation of only $k$ storage nodes, and increasing number of helper nodes, $d > k$, does not reduce the repair-cost.

The maximum degree of variables in (3.14) is $\binom{n}{k}M$. Thus, by Lemma 1, if the field size ($q$) is greater than $\binom{n}{k}M$, then there is a network coding solution for repair. Since $n_{nc} \geq 2$ (at least two coding process: one in surviving nodes, and another in the new node), $d_0 \geq \binom{n}{k}M$; thus, there is a coding solution for $q \geq d_0$.

When a node fails (assume $Q_1$), the optimization algorithm finds the minimum-cost subgraph. Accordingly, the fragments are combined using linear network coding, and then the new node is regenerated. The set including the new node ($Q'_1$) and surviving nodes must satisfy the MDS property. Thus,

$$\prod_{\{s_1, \ldots, s_{k-1}\} \subset \{2, \ldots, n\}} \det([Q'_1, Q_{s_1}, \ldots, Q_{s_{k-1}}]) \neq 0. \quad (3.15)$$

By Lemma 2, the polynomial can be nonzero. The maximum degree of each variable is less than $\binom{n-1}{k-1}Mn_{nc}$. By Lemma 1, if the finite field size $q \geq \binom{n-1}{k-1}Mn_{nc}$, there is a network solution for the repair. Clearly, $d_0 = \binom{n}{k}Mn_{nc} \geq \binom{n-1}{k-1}Mn_{nc}$ for $n \geq k$. Hence, for $q > d_0$, there exists a code for the repair. This concludes our proof. □

**Remark 3.3.** The field size in Theorem 3.1 is an upper bound for the finite field size. In practice, however, the required field size can be considerably smaller. For example, we in our recent work in [GX14], designed an explicit code for tandem and grid networks that requires than the above upper bound.

In summary, the minimum repair-cost MDS code is derived by two steps: Firstly, the optimal-cost subgraph is derived by solving a linear programming problem (3.8). Secondly, the coded fragments of the new node is regenerated by either random linear coding [HMK06] or deterministic [JSC05] from the finite field size determined by Theorem 3.1.

### 3.5 Numerical Results

In this section, we examine our proposed method on some storage networks, and show the gain of our proposed approach in reducing the repair-cost. Studies in this section will also show that the network topology can affect the repair-cost.
Optimal-cost Repair with Surviving Nodes Cooperation

3.5.1 Large Scale Tandem Storage Network

A large scale tandem storage network is illustrated in Figure 3.7. That is, each node is linked to two neighboring nodes. When a node fails and a new node joins, the repair traffic is encoded and then forwarded by the intermediate storage nodes toward the new node. In a tandem network, the optimal repair-cost is derived by the following proposition.

**Proposition 3.2.** Consider a tandem distributed storage network where each node stores $M/k$ fragments such that every $k$ nodes can reconstruct the original file of size $M$ fragments. If it is allowed to use $d \geq k$ surviving nodes in repair, then for all non-negative values of $c_{ij}$s, the optimal repair-cost is by cooperation of the $k$ closest surviving nodes to the new node and that is obtained by transmitting $M$ fragments between surviving nodes.

**Proof.** See Appendix A. \qed

This is quite a surprising result that in a tandem storage network only $k$ closest storage nodes are necessary and sufficient for the optimal repair and then increasing number of helper nodes in repair does not benefit the system. This result is in contrast to the results where the repair-bandwidth is studied [DGW+10], where the repair-bandwidth is a decreasing function of $d$ [DGW+10]. This result shows us the effect of the network topology in repair.

Figure 3.8: Large scale grid storage network. In this figure, the symbol '*' represents a transmitted fragment in repair when $d = 4$. In addition, the symbol '+' represents a transmitted fragment in repair when $d = 5$. 
3.5. Numerical Results

3.5.2 Large Scale Grid Storage Network

In a large scale grid storage network, each storage node has four neighboring nodes. This is illustrated in Figure 3.8. For this network, when there are \( d = k \) surviving nodes in the repair the optimal repair-cost is derived similar to the tandem storage network, as stated in the following proposition.

**Proposition 3.3.** Consider a grid distributed storage network where each node stores \( M/k \) fragments such that every \( k \) nodes can reconstruct the original file. Assuming the cost of transmitting a fragment from node \( i \) to node \( j \) is denoted by \( c_{ij} \), then for all non-negative values of \( c_{ij} \)s, the optimal repair-cost is achieved by cooperation of the \( k \) closest surviving nodes and that is obtained by transmitting \( M \) fragments between storage nodes.

**Proof.** The proof is similar to the proof in Proposition 3.2.

Unlike repair in tandem storage network, here by increasing \( d \), the repair cost can be decreased. We show this by the following example.

**Corollary 3.1.** In the repair of node \( n_{11} \) in Figure 3.8 by the help of nodes \( n_{21}, n_{22}, n_{23}, n_{12}, n_{13} \), the optimal-cost repair is 7 units corresponding to the minimum-cost subgraph \((\tilde{z}(21)(11), \tilde{z}(22)(12), \tilde{z}(23)(13), \tilde{z}(13)(12), \tilde{z}(12)(11)) = (1, 1, 1, 2, 2)\).

**Proof.** See Appendix B.

This shows that increasing the connectivity of storage nodes will reduce the repair-cost, e.g., in the preceding example from 8 to 7 units.

3.5.3 Fully-Connected Storage Network

In a fully-connected storage network there exists always a link between a pair of storage nodes. We show by extensive simulations that surviving node cooperation can also reduce the repair-cost in a fully connected network. We first set the parameters in the distributed storage system to \( n = 10, k = 5, M = 5 \). We assume the cost of a link in the network is a random variable that has a uniform distribution over the range \([0, \sigma]\), where \( \sigma \) changes from 1 to 10 units of cost. In this setting, we compare the optimal-cost repair when SNC is allowed (i.e., proposed scheme) with the optimal-cost in classical repair model. To find the optimal point, we use Prim’s algorithm, which has polynomial-time complexity. In Figure 3.9 we compare the optimal repair-cost in these two schemes over varying \( \sigma \). For each point on the graphs, we perform the experiment for 100 times and then evaluate the average point. Figure 3.9 shows that the proposed scheme has always a lower cost compared to the classical repair scheme. This gain of our proposed scheme increases when the link costs vary in a larger range (i.e., increasing \( \sigma \)).
Figure 3.9: Repair-cost comparison between the proposed scheme and classical repair, when $\sigma$ changes. The proposed scheme uses surviving node cooperation and outperforms the classical repair.

$\sigma$ changes, the proposed scheme uses surviving node cooperation and outperforms the classical repair.

Figure 3.10: Repair-cost comparison between the proposed scheme and classical repair, when $n$ changes. The proposed scheme has smaller repair-cost compared to classical repair for various $n$. Here, by increasing the number of nodes, the chance of having smaller cost links increases. Therefore, the repair-cost decreases by increasing $n$. 

$\rho = 10, k = 5, M = 5$
Next, we fix the parameters in the distributed storage system to $k = 5$, $M = 5$, $\sigma = 10$. We assume that the cost of a link in the network is a random variable that has a uniform distribution over the range $[0, \sigma = 10]$. In this setting, we compare the optimal-cost repair when SNC is allowed (i.e., proposed scheme) with the optimal-cost in the classical repair model. In Figure 3.10, we compare these two schemes when the number of storage nodes, $n$, changes from 1 to 10. The figure shows that the proposed scheme always outperforms the classical repair scheme. The figure also shows that by increasing $n$, the repair-cost is decreasing. The reason is that by increasing $n$, the chance of having lower cost links will increase.

### 3.6 Explicit Construction of Exact Optimal-cost Codes

In this section, we show that the lower bound of the repair cost can be achieved for the exact repair in the examples of tandem and grid networks. The code design has two steps. First, the lower bound of the repair cost and the corresponding subgraph are found by the method described in the previous section. In the second step, the exact MDS regenerating code matching to the corresponding subgraph is derived.

#### 3.6.1 Explicit Code Construction in Tandem Networks

In a tandem network, nodes are in a line topology. That is, each node is linked to two neighboring nodes, except the nodes in the line ends, which have only one neighbor. When a node fails and a new node joins, the repair traffic is encoded by the intermediate nodes and sent toward the new node. For these networks, the lower bound of the repair-cost is derived by Proposition 3.2.

Moreover, the minimum repair cost is by cooperation of the $k$ nearest surviving nodes to the new node where each node transmits $M/k$ fragments to its neighbor.

In what follows, we shall give the code construction with the optimal cost. We split the source file of a size $M$ into $k$ fragments. We denote the source file by vector $m = [m_1 m_2 \cdots m_k]^T$. To construct $(n, k)$-MDS code, we use a $k \times n$ Vandermonde matrix $G$ as a generator matrix. $G$,

$$G = \begin{pmatrix} \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{k-1} \\ \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_n & \alpha_n^2 & \cdots & \alpha_n^{k-1} \end{pmatrix}. \quad (3.17)$$

where $\alpha_i$ for $i \in \{1, \cdots, n\}$ are distinct elements from a finite field $\text{GF}(q)$.

By the property of Vandermonde matrix, every matrix constituting $k$ rows of $G$ is full rank if $\alpha_i$, $(i \in \{1, \cdots, n\})$, are distinct elements. This requires $q > n$. 

$$\Gamma \geq M. \quad (3.16)$$
Each row e.g., row $i$ ($i \in \{1, \cdots , n\}$) in matrix $G$ represents the code in node $i$. We denote the encoded data in node $i$ as $v_i$, then,

$$v_i = m_1 + m_2\alpha_i + \cdots + m_k\alpha_i^{k-1} = [1 \alpha_i \cdots \alpha_i^{k-1}]m.$$  

(3.18)

Clearly, a data collector can reconstruct the source file by connecting to any $k$ nodes. We shall further show that the exact minimum-cost repair is possible by linear codes. For illustration, we assume nodes are labeled in order, i.e., node 1 connects to node 2, node 2 connects to node 1 and 3, and so on. By Proposition 3.2, for $M = k$ fragments, the lower bound of repair-cost is by transmitting $M/k = k/k = 1$ fragment to the neighbor. Assume node $t$ ($t \in \{1, \cdots , n\}$) fails and a set of nodes $\{\text{node}_{t-k_1}, \text{node}_{t-k_1+1}, \cdots , \text{node}_{t-1}\}$ and a set of nodes $\{\text{node}_{t+1}, \text{node}_{t+2}, \cdots , \text{node}_{t+k_2}\}$ where $k_1 + k_2 = k$ help to regenerate the new node.

The repair process is as follows. The new node receives fragments from two directions, namely, aggregated data from nodes node$_{t-k_1}$ via node $t-1$, and aggregated data from nodes node$_{t+k_2}$ via node $t+1$. Thus, in one direction node node$_{t-k_1}$ multiplies its content by a coefficient $\xi_{t-k_1}$ from $GF(q)$ and sends the result to node node$_{t-k_1+1}$. Then node node$_{t-k_1+1}$ and combines the result to the received fragment and then sends its combined fragment to its next neighbor node$_{t-k_1+2}$. Finally node $t-1$ transmits the combined fragment $w_{t-1}$, which is

$$w_{t-1} = \xi_{t-k_1}v_{t-k_1} + \xi_{t-k_1+1}v_{t-k_1+1} + \cdots + \xi_{t-1}v_{t-1}$$

$$= \xi_{t-k_1}(m_1 + m_2\alpha_{t-k_1} + \cdots + m_k\alpha_{t-k_1}^{k-1})$$

$$+ \xi_{t-k_1+1}(m_1 + m_2\alpha_{t-k_1+1} + \cdots + m_k\alpha_{t-k_1+1}^{k-1})$$

$$+ \cdots + \xi_{t-1}(m_1 + m_2\alpha_{t-1} + \cdots + m_k\alpha_{t-1}^{k-1}).$$

(3.19)

In another direction, node node $t+1$ similarly sends the aggregated fragment to the new node. That is node $t+1$ transmits the combined fragment $w_{t+1}$, which is

$$w_{t+1} = \xi_{t+k_2}v_{t+k_2} + \xi_{t+k_2+1}v_{t+k_2+2} + \cdots + \xi_{t+1}v_{t+1}$$

$$= \xi_{t+k_2}(m_1 + m_2\alpha_{t+k_2} + \cdots + m_k\alpha_{t+k_2}^{k-1})$$

$$+ \xi_{t+k_2+1}(m_1 + m_2\alpha_{t+k_2+1} + \cdots + m_k\alpha_{t+k_2+1}^{k-1})$$

$$+ \cdots + \xi_{t+1}(m_1 + m_2\alpha_{t+1} + \cdots + m_k\alpha_{t+1}^{k-1}).$$

(3.20)

To achieve exact repair, we set $w_{t-1} + w_{t+1} = v_t$

$$v_t = m_1 + m_2\alpha_t + \cdots + m_k\alpha_t^{k-1}.$$  

(3.21)
Thus, vector \( \xi = [\xi_{t-k_1}, \ldots, \xi_{t-1}, \xi_{t+1}, \ldots, \xi_{t+k_2}]^T \) should be selected such that,

\[
\begin{pmatrix}
1 & 1 & \cdots & 1 \\
\alpha_{t-k_1} & \alpha_{t-k_1+1} & \cdots & \alpha_{t+k_2} \\
\alpha^2_{t-k_1} & \alpha^2_{t-k_1+1} & \cdots & \alpha^2_{t+k_2} \\
\vdots & \vdots & \ddots & \vdots \\
\alpha^{k_1}_{t-k_1} & \alpha^{k_1}_{t-k_1+1} & \cdots & \alpha^{k_1}_{t+k_2}
\end{pmatrix}
\begin{pmatrix}
\xi_{t-k_1} \\
\xi_{t-k_1+1} \\
\vdots \\
\xi_{t+k_2}
\end{pmatrix}
= 
\begin{pmatrix}
1 \\
\alpha_t \\
\alpha^2_t \\
\vdots \\
\alpha^{k_1}_t
\end{pmatrix}.
\]

(3.22)

Since matrix \( A \) is non-singular, we can select linear codes for repair at node \( t \) as \( \xi_t = A^{-1}v_t \) that makes the exact repair possible. Hence, for the successful reconstruction and repair process, a finite field \( q > n \) suffices.

### 3.6.2 Explicit Code Construction in Grid Networks

In what follows, we show that the lower bound of the repair-cost is also achievable for the exact repair in a 2 \( \times \) 3 grid network. We use systematic codes. That is, a file is equally divided to \( k \) parts (\( k \) uncoded fragments). Then, \( k \) systematic nodes store the uncoded fragments. Other parity nodes store a linear combination of the original fragments such that for any selection of \( k \) nodes, the original file can be reconstructed (MDS property). We design a code such that all systematic nodes can be exactly regenerated with the minimum cost. We note that in our codes, parity nodes can also be repaired with the minimum-cost, following the dual relationship between parity and systematic nodes. Thus, we only show the exact minimum-cost repair on systematic nodes in what follows.

Consider a distributed storage system with parameters \((n = 6, k = 3, M = 6)\), where each node stores \( M/k = 2 \) fragments, in a 2 \( \times \) 3 grid network shown in Figure 3.11. When a node fails a new node is regenerated by the remaining 5 nodes. The code has 3 nodes as systematic nodes and 3 nodes as parity nodes. By next proposition, the minimum repair-cost is 5 units. The corresponding optimal-cost subgraph for different nodes might be different. The optimal-cost repair for the repair on node 6 has been shown in the Figure 3.11.

**Proposition 3.4.** The lower bound of repair cost for nodes in the distributed storage system with parameters \((M = 6, k = 3)\), where each node stores \( M/k = 2 \) fragments, in a 2 \( \times \) 3 grid network having one unit of cost for transmitting one fragment between neighboring nodes, is by 5 units.

**Proof (sketch).** For the repair on each node, we perform \( \binom{5}{2} = 10 \) cut-set analysis on the information flow graph and then solve a linear programming problem to achieve the minimum-cost subgraphs.

For this 2 \( \times \) 3 grid network, we show the existence of linear codes for the exact repair with minimum-cost. The exact regenerating code is found by the interference alignment/cancellation techniques, as will be described later.
Code Construction

For the source file of size $M = 6$ fragments, we put the fragments into two vectors as $\mathbf{m}_1 = [a_1, b_1, c_1]$ and $\mathbf{m}_2 = [a_2, b_2, c_2]$. Every node then stores two fragments. Nodes 4, 5, and 6 are systematic nodes. That is, node 4 stores fragments $a_1$, and $a_2$, node 5 stores fragments $b_1$, and $b_2$, and node 6 stores fragments $c_1$, and $c_2$.

For parity nodes, let $p_{i1}$ and $p_{i2}$ denote two fragments stored on node $i$ for $i = 1, 2, 3$. To construct a proper code, we first construct a $3 \times 3$ Vandermonde matrix as

$$
\mathbf{v} = \begin{pmatrix}
\xi_1 & \xi_2 & \xi_3 \\
\xi_{11} & \xi_{21} & \xi_{31} \\
\xi_{12} & \xi_{22} & \xi_{32} \\
\xi_{13} & \xi_{23} & \xi_{33}
\end{pmatrix}
$$

$$
= \begin{pmatrix}
1 & 1 & 1 \\
\alpha_1 & \alpha_2 & \alpha_3 \\
\alpha_1^2 & \alpha_2^2 & \alpha_3^2
\end{pmatrix},
$$

where $\alpha_1, \alpha_2, \alpha_3$ are distinct elements in $\text{GF}(q)$. Parameter $q$ denotes the finite field size and is the design parameter. For the Vandermonde matrix there must be $q \geq 3$.

Finally, the fragments on nodes 1, 2 and 3 are coded as,

$$
\begin{pmatrix}
p_{11} \\
p_{12}
\end{pmatrix} = \begin{pmatrix}
\rho_1 a_2 + \xi_1 m_1 \\
\xi_1 m_2
\end{pmatrix},
$$

$$
\begin{pmatrix}
p_{21} \\
p_{22}
\end{pmatrix} = \begin{pmatrix}
\rho_2 b_2 + \xi_2 m_1 \\
\xi_2 m_2
\end{pmatrix},
$$

$$
\begin{pmatrix}
p_{31} \\
p_{32}
\end{pmatrix} = \begin{pmatrix}
\rho_3 c_2 + \xi_3 m_1 \\
\xi_3 m_2
\end{pmatrix},
$$

where $\rho_1, \rho_2, \rho_3$ are non-zero elements in $\text{GF}(q)$.
3.6. Explicit Construction of Exact Optimal-cost Codes

Exact repair

We show the repair process on node 6 and 5. Due to symmetry, repair in another node is similar.

- Exact repair for node 6: First, node 4 transmits fragment $a_1$ to the node 5. Then node 2 sends $p_{21}$ to the node 5. Then node 5 removes the interference $a_1, b_1, b_2$ from $p_{21}$ and sends $c_1$ to the new node. Then node 4 sends $\xi_{31}a_1 + \xi_{32}b_2$ which is aligned with the interference in $p_{31}$. Finally, the new node receives $c_1$ and $c_2 + \xi_{31}a_1 + \xi_{32}b_1 + \xi_{33}c_1$, and an interference vector $\xi_{31}a_1 + \xi_{32}b_1$. After removing interference, $c_1$ and $c_2$ are recovered in the new node.

- Exact repair on node 5: First, node 3 transmits fragment $p_{31}$ to node 6. In node 6, interference $c_1$ and $c_2$ are removed and the fragment $\xi_{31}a_1 + \xi_{32}b_2$ is transmitted to node 5. Node 5 also receives fragment $a_1$ from node 4. Then interference $a_1$ is also removed from $\xi_{31}a_1 + \xi_{32}b_1$ and $b_1$ is recovered. To recover $b_2$, node 2 transmits $p_{21}$. Then node 6 transmits $c_1$. We note that the new node already received fragment $a_1$. Thus interference $a_1$ and $c_1$ are removed from fragment $p_{21}$. By recovering $b_1$, the fragment $b_2$ is recovered from fragment $p_{21}$.

Thus, with $\rho_1, \rho_2, \rho_3 \neq 0$, and Vandermonde matrix $\xi$, we achieve exact repair with minimum-cost.

Reconstruction process

We show as follows that MDS property can be preserved if coefficients $\rho_1, \rho_2, \rho_3$ are non-zero, and matrix $\xi$ is a Vandermonde matrix. We consider the MDS property in four different cases: (1) DC connects to 3 parity nodes, (2) DC connects to 2 parity node and 1 systematic nodes, (3) DC connects to 1 parity node and 2 systematic nodes, and (4) DC connects to 3 systematic nodes.

- Case 1: There will be only one way of selecting 3 parity nodes. For the DC to recover the file, it requires to solve the following linear equations.

$$
\begin{pmatrix}
\xi_{11} & \xi_{12} & \xi_{12} & \rho_1 & 0 & 0 \\
\xi_{21} & \xi_{22} & \xi_{23} & 0 & \rho_2 & 0 \\
\xi_{31} & \xi_{32} & \xi_{33} & 0 & 0 & \rho_3 \\
0 & 0 & 0 & \xi_{11} & \xi_{12} & \xi_{12} \\
0 & 0 & 0 & \xi_{21} & \xi_{22} & \xi_{23} \\
0 & 0 & 0 & \xi_{31} & \xi_{32} & \xi_{32}
\end{pmatrix}
\begin{pmatrix}
a_1 \\
b_1 \\
c_1 \\
a_2 \\
b_2 \\
c_2
\end{pmatrix}
= r, \quad (3.26)
$$

where vector $r$ is the received vector and is known for the DC. For recovering the original file, it requires matrix $F$ to be invertible. This can be proven by the following lemma.
Lemma 3.3 (Binomial inverse theorem). For the square matrices $A$, $B$ and $C$, when $A = B + C$, then
\[
A^{-1} = (B + C)^{-1} = B^{-1} - B^{-1}(I + CB^{-1})^{-1}CB^{-1},
\]  
where $I$ is a proper identity matrix. For the case,
\[
F = \begin{pmatrix}
\xi_{11} & \xi_{12} & \xi_{13} & 0 & 0 & 0 \\
\xi_{21} & \xi_{22} & \xi_{23} & 0 & 0 & 0 \\
\xi_{31} & \xi_{32} & \xi_{33} & 0 & 0 & 0 \\
0 & 0 & 0 & \xi_{11} & \xi_{12} & \xi_{13} \\
0 & 0 & 0 & \xi_{21} & \xi_{22} & \xi_{23} \\
0 & 0 & 0 & \xi_{31} & \xi_{32} & \xi_{33}
\end{pmatrix} +
\begin{pmatrix}
0 & 0 & 0 & \rho_1 & 0 & 0 \\
0 & 0 & 0 & 0 & \rho_2 & 0 \\
0 & 0 & 0 & 0 & 0 & \rho_3 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]  
\[
(3.28)
\]

By Lemma 1, since matrices $B$ and $I + CB^{-1}$ are invertible, matrix $F$ is invertible. Thus, the DC can rebuild the source.

Case 2: DC connects to 2 parity nodes and 1 systematic node. There will be 9 scenarios of selecting 2 parity nodes among 3 parity nodes and 1 systematic node among 3 systematic nodes. We use one of them for illustration. Assume DC connects to nodes 1, 2, and 4. Thus DC has access to $a_1$, $a_2$. To recover the original file DC solves the following linear equation.
\[
\begin{pmatrix}
\xi_{12} & \xi_{13} & 0 & 0 \\
\xi_{22} & \xi_{23} & \rho_2 & 0 \\
0 & 0 & \xi_{12} & \xi_{13} \\
0 & 0 & \xi_{22} & \xi_{23}
\end{pmatrix}
\begin{pmatrix}
b_1 \\
c_1 \\
b_2 \\
c_2
\end{pmatrix} = r',
\]  
\[
(3.29)
\]

where vector $r'$ is known for DC. Using Lemma 1, the linear equation has unique solution and then DC recovers the original file.

Case 3: DC connects to 1 parity node and 2 systematic nodes. For this scenario, DC recovers the original file with given Vandermonde matrix $\xi$ and $\rho_1, \rho_2, \rho_3 \neq 0$. 
3.7 Conclusions

Case 4: DC connects to 3 systematic nodes. In this scenario, it is straightforward for DC to recover the source.

Hence, above codes have minimum-cost exact repair with a finite field size $q \geq 3$.

3.7 Conclusions

We studied the repair-cost in distributed storage systems where storage nodes are connected together by an arbitrary network topology. We proposed the optimal codes that exploit the multi-hop network structure in repair. We formulated a linear programming problem which gives the fundamental lower-bound of the repair-cost. The linear programming also leads us to the optimal-cost policy for MDS codes. We proved the existence of the code by the random linear code over a large finite field size. We discussed the required field size for the existence of the code. To reduce the cost in networks, we proposed the surviving node cooperation approach. We examined our proposed method over some large-scale storage networks and then presented the advantages in deploying the optimal-cost repair. We proposed explicit constructions for exact repair in two networks. Studying the repair in networks having the constraint on bandwidth and delay in fragment transmissions is interesting and can be followed as future works.

3.8 Appendices

3.8.1 Proof of Proposition 3.2

Assume $d \geq k$ surviving nodes are helping a new node to be regenerated. Nodes are labeled from $d$ to 1 as shown in Figure 3.12. For the convenience, let us assume that node 1 fails and then consecutive nodes 2 till $k + 1$ are the $k$ closest nodes to the new node (1’), which is located at the same position as node 1. We derive the constraint region in a general form and then find the minimum repair-cost. We obtain the constraint region by the following steps:

Step 1: We connect DC to the new node and nodes $k - 1$ other nodes in the set of helper nodes $\mathcal{H} = \{2, 3, \ldots, d + 1\}$. Then the cut constraint is,

$$z_{(21')} + (k - 1)\alpha \geq M \Rightarrow z_{(21')} \geq M/k.$$  \hfill (3.30)

Step 2: We connect DC to the new node, node 2 and $d - 2$ other nodes in $\mathcal{H} = \{2, 3, \ldots, d + 1\} - \{i\}$, for $i \in \{3, \ldots, k + 1\}$. Then the cut that passes the edge $(i, i - 1)$ gives the following constraint:

$$z_{(i,i-1)} + (k - 1)\alpha \geq M \Rightarrow z_{(i,i-1)} \geq M/k,$$

for $i \in \{3, \ldots, k + 1\}$. \hfill (3.31)

Step 3: We connect DC to the new node, node 2, $\ldots, k + 1$. Then the cut constraint is,
Figure 3.12: Cut analysis in a tandem network. The cut analysis in this figure corresponds to the constraint $z_{21'} + (k - 1)\alpha \geq M$.

$$z_{(j,j-1)} + k\alpha \geq M \Rightarrow z_{(j,j-1)} \geq 0$$

for $j \in \{k + 2, k + 3, \ldots, d\}$. \hspace{1cm} (3.32)

This shows $k$ closest neighbour must individually transmit $M/k$ fragments toward the new node. Hence, whatever the value of $d$ is, the $k$ closest neighbours to the new node must totally transmit $k \times M/k = M$ fragments, which poses the cost of $\sum_{i=2}^{k+1} c_{(i,j-1)} M/k$ units in repair. Thus, increasing $d$ more than $k$ does not reduce the repair-cost.

### 3.8.2 Proof of Corollary 3.1

The corresponding cost vector ($c$) for the repair on node 6 is as below. We formulate the linear optimization problem as follows. There are $\binom{d-1}{k-1} = 10$ cut constraints. Figure 3.13 shows one of these cut constraints. For convenience let call nodes $n_{21}, n_{22}, n_{23}, n_{13}, n_{12}, n_{11}$ respectively as nodes 1, 2, 3, 4, 5, 6. Then we have,

$$c = [c_{(12)}c_{(23)}c_{(14)}c_{(25)}c_{(36)}c_{(56)}]^T = 1.$$ \hspace{1cm} (3.33)

$$\begin{array}{l}
\min \Gamma(z) \\
\text{s.t.} \quad Hz \geq (M - 3\alpha)1 \\
\quad z_{ij} \geq 0,
\end{array} \hspace{1cm} (3.34)$$
3.8. Appendices

Figure 3.13: Cut analysis in a $2 \times 3$ grid network with $n = 6$, $k = 4$. The cut in this figure corresponds to the constraint $z_{(56)} \geq M - 3\alpha$.

where $H =$

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 
\end{bmatrix}
$$

and $z =$

$$
\begin{bmatrix}
z_{12} \\
z_{14} \\
z_{23} \\
z_{25} \\
z_{36} \\
z_{45} \\
z_{56}
\end{bmatrix}
$$

We note that matrix $H$ is resulted from cut-set analysis in the first stage of repair. For example, the cut mentioned in Figure 3.13 constructs the 8-th row in matrix $H$, corresponding to inequality $z_{(56)} \geq M - 3\alpha$.

Solving this linear optimization problem (e.g. by the simplex method) for the $M = 8$ and $\alpha = 2$ results in:

$\Gamma = 7$, and

$z = (z_{(12)}, z_{(14)}, z_{(23)}, z_{(25)}, z_{(36)}, z_{(45)}, z_{(56)}) = (0, 1, 0, 1, 2, 2)$.
Part II

Distributed Storage Systems with Packet Erasure Channels
Chapter 4

Repair for Distributed Storage Systems with Packet Erasure Channels and Dedicated Nodes for Repair

We in this chapter study the repair problem in distributed storage systems where storage nodes are connected through packet erasure channels and some nodes are dedicated to repair (termed as dedicated-for-repair (DR) storage nodes). We first investigate the minimum required repair-bandwidth in an asymptotic setup, in which the stored file is assumed to have an infinite size. The result shows that the asymptotic repair-bandwidth over packet erasure channels with a fixed erasure probability has a closed-form relation to the repair-bandwidth in lossless networks. Next, we show the benefits of DR storage nodes in reducing the repair bandwidth, and then we derive the necessary minimal storage space of DR storage nodes. Finally, we study the repair in a non-asymptotic setup, where the stored file size is finite. We study the minimum practical-repair-bandwidth, i.e., the repair-bandwidth for achieving a given probability of successful repair. A combinatorial optimization problem is formulated to provide the optimal practical-repair-bandwidth for a given packet erasure probability. We show the gain of our proposed approaches in reducing the repair-bandwidth.

4.1 Introduction

Distributed storage systems have recently attracted significant research interests for many applications in data centers and peer-to-peer storage networks, e.g., OceanStore, Total Recall and DHash++ [DGW+10]. While existing research results are mostly for distributed storage systems over wired networks, distributed storage systems can also be applied over wireless networks. The use of wireless transmission between storage nodes has been suggested in [CWCC11] to combat congestion (oversubscription) in data centers. Distributed storage systems can also be used in wireless caching networks with device-to-device communica-
tion [GXS15, GXSS15, PHT13, HKBL14]. These systems can also have applications in delay tolerant networks (DTNs) [GX13].

In distributed storage systems, node failure and repair have been studied [DGW+10, WD09, KLSS11, RSRK12, SRKR12, GX13, GXS11, GXS15, GXSS15, PHT13, HKBL14]. More specifically, if a storage node fails or leaves the system, in a mechanism, called repair, a new node is regenerated by transmitting sufficient data from the surviving nodes to the new node. In the most of the existing results, it is assumed that the links between storage nodes are error-free, without any error or erasure during repairing information transmission. However, in the distributed storage systems over wired networks, the transmitting packets may be lost due to e.g., congestion and buffer overflow in intermediate switches/routers, or due to protocol and load balancer issues, as it is reported in [GJN11] for long-term measurement analysis over the links in the current data-centers. In wireless networks, repairing packets may be lost due to channel fading or interference [Mad08, AZBS06].

Based on above observations, we will study the repair problem in a distributed storage system with lossy channels. We derive the optimal storage-bandwidth trade-offs for distributed storage systems with packet erasure channels. We show that the optimal tradeoffs are asymptotically achievable (when the file size is infinitely large). More specifically, consider a source file that contains $M$ fragments, where each fragment contains $\xi$ packets (thus, the file contains $M \xi$ packets). Suppose that the surviving nodes totally transmit $\gamma'(\xi)$ packets to the new node. Asymptotic analysis gives the minimum bandwidth $\frac{\gamma'(\xi)}{M \xi}$ when $\xi$ tends to infinity, which may be used as the performance bound for a finite $\xi$.

As node failure in distributed storage systems may frequently happen [DRWS11, SAP+13], it is valuable to have some nodes dedicated to repair to reduce the repair bandwidth. We term these nodes dedicated-for-repair (DR) nodes. We investigate the benefits of DR storage nodes. For the storage nodes participating both file recovery and repair, we call them the complete storage nodes. In a distributed storage system, the repair-bandwidth could be reduced by using more complete storage nodes in repair [DGW+10], if the system allows. However, in some scenarios, the system may not allow to increase the number of complete storage nodes in repair due to e.g., limitations in storage capacity and the repair-bandwidth. Yet, if we have additional storage nodes with smaller storage capacity and smaller repair-bandwidth (than complete storage node) and they function the same as complete storage nodes for repairing, then the repairing costs could be saved for storage space and repair-bandwidth. Note that DR storage nodes are designated only for repair. That is, DR nodes will not participate source file recovery by data collectors. We will study the minimal storage space of DR storage nodes and will show the gain of using DR storage nodes in reducing the repair bandwidth.

Though asymptotic analysis can serve as performance bounds and is valid especially for very large file size, in practice the file size might be limited, namely, a finite number of repairing packets. For this case, we show that the repair-bandwidth can

$^1\gamma'$ is a function of $\xi$.\)
4.2 Background and Related Works

In a distributed storage system, when a node fails, to maintain the reliability of the system, a node is regenerated. In the regenerating process, the surviving nodes transmit sufficient data to the new node such that the system with the new node still maintains the reconstruction property (any $k$ out of $n$ complete storage nodes can reconstruct the stored file). Yet, the new node may have different coded packets compared to the failed one. This is termed as functional repair. On the other side, for exact repair, the coded packets of the new node are exactly the same as those in the failed node. We mainly consider functional repair.

Reference [?] modeled the repair process by an information flow graph, and mapped the repair problem into a multicast problem in lossless networks (no transmission errors). Cut-set bound analysis on the information flow graph showed that a sink (or a data collector) can reconstruct the original file of size $M\xi$ packets if and only if

$$\sum_{i=0}^{k-1} \min\{\alpha \xi, (d - i)\beta \xi\} \geq M \xi, \quad (4.1)$$

where each storage node stores $\alpha$ fragments such that any set of $k$ storage nodes can reconstruct the original file, and in repair $d$ ($d \geq k$) number of surviving nodes transmit $d\beta$ fragments to the new node. The above fundamental bound leads us to the capacity of such systems. In what follows, we will provide two useful definitions in distributed storage systems.
**Definition 4.1** (Information Rate in a Distributed Storage System). The information rate in a distributed storage system under the dynamic of node failure/repair is defined as the amount of information that a data collector obtains by connecting to \( k \) or more (complete) storage nodes in one unit of time.

Here, the dynamic of node failure/repair means the process in which (different) nodes continuously fail and are repaired.

**Definition 4.2** (Capacity of a Distributed Storage System). The maximum amount of information that a data collector can obtain in one unit of time by connecting to \( k \) or more (complete) storage nodes is defined as the capacity. The term \( \sum_{i=0}^{k-1} \min\{\alpha \xi, (d - i) \beta \xi\} \) equals to the capacity of a distributed storage system in lossless networks \( [DGW+10] \). For the optimal codes, \( \sum_{i=0}^{k-1} \min\{\alpha \xi, (d - i) \beta \xi\} = M \xi \).

The authors in \( [DGW+10] \) derived an explicit relation between \( \alpha, \gamma, d, M \) and \( k \) for the optimal storage-bandwidth tradeoff. The codes achieving the optimal tradeoff are called regenerating codes. The codes achieving two extreme points on the optimal storage-bandwidth tradeoff are called minimum storage regenerating (MSR) and minimum bandwidth regenerating (MBR) codes, respectively. The optimal repair-bandwidth at these two extreme points can also be derived by two sequential optimization processes under the constraint (4.1). For the optimal MSR codes, we can derive the minimum repair-bandwidth for an MSR code as

\[
\alpha_{\text{MSR}} = \frac{M}{k}, \\
\gamma_{\text{MSR}} = \frac{Md}{k(d-k+1)}. \tag{4.2}
\]

In the optimization process, if we first minimize the repair-bandwidth and then storage per node, the other extreme point, namely, MBR is achieved. It can be easily verified that in general \( \gamma \geq \alpha \) \( [SRKR12] \). For the MBR codes \( \gamma = \alpha \). Therefore, setting \( \gamma = d \beta = \alpha \) on the optimum bound \( \sum_{i=0}^{k-1} \min\{\alpha, (d - i) \beta\} = M \) yields

\[
\alpha_{\text{MBR}} = \frac{2Md}{k(d-k+1)}, \\
\gamma_{\text{MBR}} = \frac{2Md}{k(d-k+1)}. \tag{4.3}
\]

The code construction and the achievability of the functional and exact repair have been studied in \( [Wu10b, WD09, RSKR09b] \). In \( [JFP04] \), cooperative regenerating codes are proposed to reduce the bandwidth in the scenario of multiple-node failure. Surviving node cooperation is suggested in \( [GXSL11, GXSFS13] \) in order to minimize the cost of repair in multi-hop storage networks. Yet, in most of the previous works of regenerating codes, it is assumed that the links between storage nodes
are perfect. That is, there is no error or erasure. Recently Rashmi et al. [RSRK12] proposed a regenerating code which is resistant to a specific number of path failures by requesting more nodes to join the repair process. Particularly, for the code resistant to $d_2$ path failures, it requires to transmit from $d_{\text{tot}} = d_1 + d_2$ surviving nodes instead of $d_1$ nodes ($d_1$ nodes are assumed to be sufficient for repair with perfect channels). Compared to [RSRK12], we consider the probability of successful repair. A regenerating process is successful when the new node together with the surviving nodes has the reconstruction property. We show that optimal $d_1$ and $d_2$, in which the probability of successful repair is maximized, depend on the erasure probability of the links. In addition, we study the capacity of distributed storage networks where the channels between the surviving nodes and the new node are packet erasure channels, and propose a method to reduce the repair-bandwidth by DR storage nodes.

Other related results on network coding for erasure networks are as follows. In [LMKE08], it is shown that random linear codes can achieve the capacity of packet erasure networks. The capacity of wireless erasure networks has been studied in [DGP+06]. In [JDPF05], [LDH09], the probability of successful reconstruction of a source file is studied in packet erasure networks. In contrast to [JDPF05], [LDH09], we study the repair problem.

### 4.3 System Model

Consider that a distributed storage system stores a file of $M$ fragments, where each fragment contains $\xi$ packets. We refer to $\xi$ as subpacketization order. We assume that repairing information from the surviving nodes to the new node is packetized. A packet here is considered as the basic information unit and is denoted by a vector of $\rho$ symbols taken from the finite field $GF(q)$, where $q$ is the alphabet size. Hence, a packet has $\rho \log_2(q)$ bits of information [LMKE08]. We assume that channel coding has been used for each packet and perfect error detection is also used. Therefore, a packet in any channel is either received error-free or is dropped and each link can be regarded as a packet erasure channel [DGP+06]. When a packet has only one bit of information, then the channel can also be regarded as a binary erasure channel. We also assume packets are erased independently with a given probability $p$ and channels are memoryless. We remark that all the communication channels are point to point, and thus the effect of broadcast and interference in wireless channels are not studied here.

In our system, there are $n$ storage nodes with a storage capacity $\alpha$ fragments and every $k$ nodes can reconstruct the source file. Moreover there are $h(h \geq 0)$ DR storage nodes, each of which with storage capacity $\alpha'$ fragments. To repair a failed node, $d(d \geq k)$ surviving nodes among $n - 1$ complete storage nodes plus $h$ DR storage nodes equally send $\gamma \xi = (d + h)\beta \xi$ packets (in total) to the new node. For simplifying illustration, we denote the distributed storage system with packet erasure channels by DSS($n, k, d, h, \alpha, \alpha', \gamma = d\beta, M\xi, p$). We note that, in
Figure 4.1: Information flow graph for distributed storage systems with packet erasure channels and DR storage nodes. CS stands for a complete storage node and DR stands for a DR storage node.

this system, the reconstruction property is relaxed for DR storage nodes. That is, $k$ storage nodes including at least one DR storage node cannot reconstruct the original file.

We extend the regenerating codes in two senses: firstly we extend the regenerating code to lossy networks, and secondly we extend the regenerating code by adding the DR storage nodes to the repair process. Therefore, it would be useful to define the extended MSR (EMSR) and extended MBR (EMBR) codes as follows.

**Definition 4.3 (EMSR$_{h,p}$ Codes).** Consider a distributed storage system that stores a file of size $M$ fragments by an $(n,k,d)$ – EMSR$_{h,p}$ codes. Then each complete storage node stores $\alpha = M/k$ fragments, and every $k$ out of $n$ complete storage nodes can reconstruct the source file. In repair, $d$ complete storage nodes plus $h$ DR storage nodes help generating a new node. The repair packets over the channels between the helper nodes and the new node are lost with a probability $p$. These codes have the minimum repair-bandwidth for the extended regenerating codes where $\alpha = M/k$.

**Definition 4.4 (EMBR$_{h,p}$ Codes).** Consider a distributed storage system that stores a file of size $M$ fragments by an $(n,k,d)$ – EMBR$_{h,p}$ codes. Then each complete storage node stores $\alpha > M/k$ fragments, and every $k$ out of $n$ complete storage nodes can reconstruct the source file. In repair, $d$ complete storage nodes plus $h$ DR storage nodes help generating a new node. The repair packets over the channels between the helper nodes and the new node are lost with a probability $p$. These codes have the minimum repair-bandwidth among all the extended regenerating codes.
4.4 Asymptotic Analysis

4.4.1 Information Flow Graph

To study the fundamental performance limits, we will first give the analysis when the file size is asymptotically large. Our analysis is based on a graphical representation of a distributed storage system, namely, the information flow graph $\mathcal{G}$, which was first adopted in [DGW+10] to analyze the repair problem in lossless networks. We modify the graph in [DGW+10] to analyze the repair problem in a distributed storage system with packet erasure channels and in the presence of DR storage nodes. Then the information flow graph is a directed acyclic graph denoted by $\mathcal{G}(\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ denotes the set of nodes and $\mathcal{A}$ denotes the set of edges in the network. Each complete storage node is modeled by two nodes, $\text{in}$ and $\text{out}$ nodes, which are connected by a link of capacity $\alpha\xi/\tau$ packets per unit of time. The source is connected to $\text{in}$ nodes of complete storage nodes by links of capacity $\alpha\xi/\tau$ packets per unit of time. Each DR storage node is modeled by one node. The source is connected to a DR storage nodes by a link of capacity $\alpha\xi/\tau$ packets per unit of time. When a node fails, $d$ complete storage nodes plus $h$ DR storage nodes transmit repairing packets to the new node. Each of $d + h$ helper nodes sends packets to the new node by a rate $\beta\xi/\tau$ packets per unit of time. The transmitted packets on the lossy links might be erased with a packet erasure probability $p$. In such a graph, a data collector (DC) connects to $k$ out nodes of complete storage nodes and reconstructs the original file in $\tau$ units of time. We say that repair is successful if a data-collector by connecting any $k$ out nodes of complete storage nodes can reconstruct the original file in $\tau$ units of time. In asymptotic analysis, $\xi$ and $\tau$ tend to infinity. An information flow graph with a number of node failure/repair is shown in Fig. 4.1. We note that in the modified information flow graph we introduced the time, $\tau$ for analysis, in which data-collectors reconstruct the file. This is not required in lossless networks in [DGW+10].

A cut-set bound analysis over the above information flow graph leads us to the following upper bound of the information rate. We later discuss about the achievable bounds.

**Lemma 4.1.** An upper bound of the information rate in the above information flow graph is

$$\sum_{i=0}^{k-1} \min\{\alpha\xi/\tau, (d - i + h)\beta(1 - p)\xi/\tau\}. \quad (4.4)$$

**Proof.** The proof is inspired by the cut-set bound analysis in [DGW+10] and the cut analysis in network information theory [CT12]. The detailed proof can be found in Appendix A.
4.4.2 Optimal Storage-Bandwidth Tradeoff in Erasure Networks

In what follows, we will first investigate the fundamental performance limits on the repair in packet erasure networks for $h = 0$. In the next section, we will discuss more details about DR storage nodes and then extend the results for $h > 0$. For $h = 0$, when a node fails, a new node is generated by the help of $d$ surviving nodes (among $n - 1$ complete storage nodes). Each of $d$ surviving nodes transmits packets by a rate $\beta \xi / \tau$. The transmitted packets from the surviving nodes to the new node might be erased with an erasure probability $p$. We note that we restrict our analysis to the case where all the links have equal packet erasure probability, $p$. Where the links between $d$ surviving nodes and the new node have different packet erasure probabilities (like in heterogeneous networks), then the approach that requires $d$ nodes equally transmit $\beta$ fragments to the new node is suboptimal. The following theorem states the capacity of our distributed storage systems.

**Theorem 4.1.** Consider a distributed storage system $DSS(n, k, d, h = 0, \alpha, \alpha', \gamma = d\beta, M\xi, p)$ with a finite number of node failure/repair processes. Suppose $T$ denotes the set of data-collectors in the system, and $\tau$ denotes the units of time for reconstructing the original file by data-collectors. There exists a linear code over $\mathbb{GF}(q)$ and for $\delta > 0$ and $q > |T|/\delta$ such that data-collectors can achieve, with arbitrarily small error probability, the rate

$$R_0 = (1 - \delta) \sum_{i=0}^{k-1} \min\{\alpha \xi / \tau, (d - i)(1 - p)\beta \xi / \tau\}. \quad (4.5)$$

Hence, for large $q$ and $\xi$, and $\delta$ close to zero, there exist linear codes if storage capacity of each node, $\alpha$, is greater than $\alpha^*(n, k, d, \gamma)$. It is information-theoretically impossible to find a code for $\alpha < \alpha^*(n, k, d, \gamma)$. The function of normalized $\alpha^*(n, k, d, \gamma) (\alpha^*/M)$ over normalized $\gamma (\gamma/M)$ is given as follows.

$$\frac{\alpha^*}{M} = \begin{cases} \frac{1 - g(i)(1 - p)\gamma}{k - i} & \text{if } \frac{\gamma}{M} \in \left[0, h(0), +\infty\right), \\ \frac{\gamma}{M} & \text{if } \frac{\gamma}{M} \in \left[h(i), h(i - 1)\right), \end{cases}$$

where $i = 1, ..., k - 1$, and

$$h(i) \triangleq \frac{2d}{[(2k - i - 1)i + 2k(d - k + 1)](1 - p)}, \quad (4.6)$$

and

$$g(i) \triangleq \frac{(2d + 2k + i + 1)i}{2d}. \quad (4.7)$$

The complete proof of Theorem 4.1 is provided in Appendix B. We use infinite length codewords (infinite number of packet transmissions) to achieve the optimal
4.4. Asymptotic Analysis

Figure 4.2: Fundamental storage-bandwidth tradeoffs in packet erasure networks, for $n = 10$, $k = 5$, $d = 9$, $h = 0$, and for different packet erasure probabilities. The higher packet erasure probability ($p$), the higher repair-bandwidth.

storage-bandwidth tradeoff. We note that infinite length codewords might not be practical, as it may lead to infinite delay in repair. However the analysis is still useful since it provides us insights on the repair problem in packet erasure networks and it provide performance bounds. The result of asymptotic analysis shows a closed-form relation between the storage-bandwidth tradeoff in packet erasure networks with the tradeoff in lossless networks. That is, for a given packet erasure probability $p$ and for a given storage capacity $\alpha$, the asymptotic repair-bandwidth in the packet erasure network is $1/(1-p)$ times larger than the corresponding repair-bandwidth in the lossless network. Fig. 4.2 shows the optimal storage-bandwidth tradeoffs in packet erasure networks with $p = 0.1, 0.2, 0.3$. Expectedly, a larger packet erasure probability leads to a higher repair traffic. Two extreme points on the storage-bandwidth tradeoff can be computed by the following equations. For the EMSR codes,

$$\alpha_{\text{EMSR}} = \frac{M}{k},$$
$$\gamma_{\text{EMSR}} = \frac{Md}{k(d-k+1)(1-p)}.$$  \hspace{1cm} (4.8)

For the EMBR codes,

$$\alpha_{\text{EMBR}} = \frac{2Md}{k(2d-k+1)},$$
$$\gamma_{\text{EMBR}} = \frac{2Md}{k(2d-k+1)(1-p)}.$$  \hspace{1cm} (4.9)
4.5 Reducing Repair-Bandwidth by DR Storage Nodes

In what follows, we will discuss more details on DR storage nodes and their benefits in repair. We first discuss the benefits of DR storage nodes in lossless networks (i.e., for $p = 0$). Then we will extend the results to a general form ($p > 0$).

4.5.1 Motivation

We illustrate the use of DR storage nodes by two examples.

Example 4.1 (DR Storage Nodes for EMSR Codes). Consider a distributed storage system depicted in Figures 4.3-4.5. Suppose, a file of size 4 fragments is encoded by a $(3, 2)$-MDS code over GF(13), and distributed among 3 complete storage nodes such that any complete storage node stores 2 fragments. When there is no a DR storage node, if any storage node fails, then repairing a failed node requires 4 fragment transmissions (substituting $d = n - 1 = 2, k = 2, M = 4$ in (4.2)). Now, assume that there exists a DR storage node with the storage capacity of one fragment and storing a coded fragment $a_1 + a_2 + b_1 + b_2$. We describe how the DR storage node help in repairing nodes 1, 2 and 3. If node 1 fails, nodes 2 sends $a_2 + b_2$, node 3 sends $8a_2 + 8b_2 + 12a_1 + 9b_1$, and the DR storage node transmits $a_1 + a_2 + b_1 + b_2$, as shown in Figures 4.3-4.5. The new node can generate fragments $a_1, b_1$ by some linear operations on the received fragments. This is accomplished by removing the interfering term $a_2 + b_2$ and then solving a two-dimensional linear equation (for more details about the method of interference alignments in repair please see [WD09, SRL13, CJM13]). Similarly if nodes 2 or 3 fails, we can show that with the DR
4.5. Reducing Repair-Bandwidth by DR Storage Nodes

Figure 4.4: The DR storage node helps the repair of nodes 2.

storage node, repairing any failed node requires 3 fragment transmissions. In node 3, the fragments in the new node is generated by exact repair through these linear operations over the new node’s received fragments: $3a_1 + 2a_2 = (4a_1 + b_1) + (3a_2 + b_2) - (a_1 + b_1 + a_2 + b_2); 9b_1 + 8b_2 = 12(a_1 + b_1 + a_2 + b_2) - 3(4a_1 + b_1) - 4(3a_2 + b_2)$. Thus, in each repair process, one unit of (fragment) transmission is saved.

Example 4.2 (DR Storage Nodes for EMBR Codes). Consider a distributed storage system depicted in Figure 4.6. Here, the distributed storage system stores a file using a $(4, 3, 3)$ − EMBR$h=1, p=0$ over GF(2). A source file contains nine fragments $c_1, \ldots, c_9$. A fragment $c_{10}$ is encoded by $c_{10} = c_4 + c_7 + c_9$. Nodes 1, 2, 3 and 4 store the source file such that any 3 nodes can reconstruct the source file. A DR storage node storing $c_4, c_7, c_9$ can help all the other storage nodes in repair. This is shown in Figures 4.6-4.9.

In general, solving (4.4) in Lemma 4.1 for a given $M$ and $p=0$ leads to a lower bound of repair-bandwidth. For two extreme points, the lower bound of the repair bandwidth are as follows. For EMSR codes, we have,

$$\gamma_{EMSR_{h,p=0}} = \frac{M(d + h)}{k(d - k + h + 1)} \quad (4.10)$$

And for EMBR, we have

$$\alpha_{EMBR_{h,p=0}} = \frac{2M(d+h)}{k(2(d+h)-k+1)} \quad (4.11)$$

$$\gamma_{EMBR_{h,p=0}} = \frac{2M(d+h)}{k(2(d+h)-k+1)} \quad (4.12)$$
Figure 4.5: A distributed storage system with a DR storage node. Nodes store a file based on a\((3, 2, 2) – \text{EMSR}_{h=1,p=0}\) code. The DR storage nodes helps the repair of nodes 1, 2, and 3 respectively in Figures 4.3-4.5. The finite field here is GF(13). This example shows a DR storage node, with a half storage space compared to a complete storage node, functions similar to a complete storage node in repair.

The above results show potential reduction in the repair-bandwidth by a factor of\(d(d + h - k + 1)/((d + h)(d - k + 1))\) for EMSR codes, and by a factor of\(d(2(d + h) - k + 1)/((d + h)(2d - k + 1))\) for EMBR codes. We note that we could achieve these gains by adding \(h\) complete storage nodes. However DR storage nodes have smaller storage space (we will show later) and require less repair-bandwidth compared to complete storage nodes. The minimal \(c'\) will be discussed as follows.

4.5.2 Minimum DR Storage Capacity for the EMSR Codes

We show that there exist linear codes for repairing in a DSS satisfying the bound in (4.10) when each complete storage node stores \((\alpha = M/k)\) fragments, and each DR storage node stores \(\alpha' = \beta\) fragments.

**Lemma 4.2** (Achievability for EMSR codes with \(\alpha' = \beta\)). For a repair process in a DSS with parameters \((n, k, d, h \leq [M/\beta], \alpha = M/k, \alpha' = \beta, \beta, M)\) of EMSR codes, there exist linear codes if each of \(d + h\) helper nodes, including the DR storage nodes, transmit \(\beta = M/(k(d + h - k + 1))\) fragments to the new node. In addition,
4.5. Reducing Repair-Bandwidth by DR Storage Nodes

The functional repair is always possible after any number of node failure/repair processes.

Proof. See Appendix C.

Using Lemma 4.2 and the fact that we design $\alpha' \geq \beta$, we can deduce the optimal bound of $\alpha'$ for EMSR codes.

**Theorem 4.2.** For a repair process in a DSS with parameters $(n, k, d, h \leq \lfloor M/\beta \rfloor, \alpha = M/k, \alpha', \beta, M)$ of EMSR codes, the optimal storage space for DR storage nodes is $\alpha' = \beta$.

Theorem 4.2 shows that reducing the repair-bandwidth for the EMSR codes can be achieved by adding a DR storage node with storage capacity of $\beta$ instead of using a complete storage node with capacity $\alpha = M/k = \beta(d + h - k + 1)$. Hence, it requires less storage space by the ratio of $(d + h - k + 1)$. 

---

**Figure 4.6:** A DR storage node helps the repair of nodes 1.
Figure 4.7: The DR storage node helps the repair of nodes 2.

In the next subsection, we investigate the minimum storage required for a DR storage node for EMBR codes.

4.5.3 Minimum Repairing Storage Capacity for EMBR Codes

A DR storage node requires more storage space for EMBR codes compared to EMSR codes (where it was $\alpha' = \beta$). We will present in the following lemma that for the EMBR codes it is necessary to have $\alpha' \geq k\beta$.

**Lemma 4.3** (Non-achievability for EMBR codes with $\alpha' < k\beta$). For the EMBR codes in a DSS with parameters $(n, k, d, h, \alpha = (d + h)\beta, \alpha', \beta, M)$, $\alpha' \geq k\beta$.

*Proof.* See Appendix D.

The next lemma shows that it is sufficient for the DR storage node to have $\alpha' = k\beta$.

**Lemma 4.4** (Achievability for EMBR codes with $\alpha' = k\beta$). For the repair process of EMBR codes with DR storage nodes, each of which has storage capacity $\alpha' = k\beta$, there exists a linear code if each of $d + h$ nodes, including $h \leq \lfloor M/k\beta \rfloor$ DR storage
4.5. Reducing Repair-Bandwidth by DR Storage Nodes

Figure 4.8: The DR storage node helps the repair of nodes 3.

nodes, transmits $\beta = 2M/(k(2(d+h)-k+1))$ fragments to the new node. In addition, the functional repair is always possible after any number of node failure/repair processes.

Proof. See Appendix E.

Using Lemma 4.3 and 4.4, we can deduce the optimal bound of $\alpha'$ for EMBR codes.

**Theorem 4.3.** For a repair process in a DSS with parameters $(n,k,d,h \leq \lceil M/k\beta \rceil, \alpha,\alpha',\beta,M)$ of EMBR codes, the optimal storage space for DR storage nodes is $\alpha' = k\beta$.

Theorem 4.3 shows that reducing the repair-bandwidth for the EMBR codes can be achieved by adding a DR storage node with storage space of $\alpha' = k\beta$ instead of $\alpha = (d+h)\beta$. Note that in some scenarios (if not most), $d = n - 1$. Thus, using DR storage nodes can reduce storage space.

### 4.5.4 Repairing a DR Storage Node

Since DR storage nodes can also fail, we investigate the repair-bandwidth of DR storage nodes. We first show the repair of a DR storage node in Figure 4.10 for
Figure 4.9: A DR storage node when nodes 1, 2, 3, 4, and 5 store a file based on a (4, 3, 3) − EMBR$h=1,p=0$. The DR storage nodes help the repair of nodes 1, 2, 3, 4 respectively in (a), (b), (c), (d). The finite field here is GF(2). This example shows a DR storage node, with 3/4 storage space of a complete storage node, functions similar to a complete storage node in repair.

Similar to a complete storage node, the repair for a DR storage node can be functional or exact. In the functional repair (which includes the exact repair also) of a DR storage node, a number of complete storage nodes plus the surviving DR storage nodes transmit sufficient data to the new DR storage node. Let $\gamma_R = (d_R + h_R)\beta_R$ be the total repair bandwidth for a DR storage node, where $d_R$ denotes the number of complete storage nodes in repair, $h_R$ denotes the number of DR storage nodes (among $h - 1$ surviving DR nodes) in repair, $\beta_R$ denotes the number of fragments each helper node sends in repair. At this stage, the optimal repair-bandwidth of a DR storage node is still not known. We will show in the following proposition that the bandwidth $\gamma_R = k\beta$ is sufficient for the (exact) repair of a DR...
storage node. We note that $\gamma_R = k\beta$ may be suboptimal, but it still requires less repair-bandwidth than a complete storage node.

**Proposition 4.1.** Repair-bandwidth $\gamma_R = k\beta$ is sufficient for regenerating a new DR storage node for EMSR and EMBR codes.

**Proof.** See Appendix F.

![Diagram](image)

Figure 4.10: Repair of the DR storage node for EMSR codes. The new DR storage node is generated by transmitting two fragments from nodes 1 and 2. The repair-bandwidth of a DR storage node is smaller than the repair-bandwidth for a complete storage node.

Proposition 4.1 shows that the repair-bandwidth for a DR storage node is not greater than the repair-bandwidth for a complete storage nodes ($k\beta \leq d\beta$).

### 4.5.5 DR Storage Nodes for non-EMSR and non-EMBR Codes

In the previous subsections, we studied the DR storage node for two extreme points on the storage-bandwidth tradeoff, namely EMSR and EMBR codes. Using the same approach, we can extend the results to all the points on the storage-bandwidth tradeoff. A formal result is given in the following conjecture.

**Conjecture 4.1.** Consider a distributed storage system $(n, k, d, h, \alpha, \alpha', \beta, M, p = 0)$ with a code on the optimal storage-bandwidth tradeoff point ($\sum_{i=0}^{k-1} \min\{\alpha, (d + h - i)\beta\} = M$). Let the points on the optimal tradeoff that are between EMSR and EMBR (two extreme) points are denoted as an the interior points. If we present an interior point on the storage-bandwidth tradeoff by subdividing the set of points on the optimal storage-bandwidth tradeoff into $(k - 1)$ subsets as follows: $S_\theta =$
Figure 4.11: Repair of the DR storage node for EMBR codes. The new DR storage node is generated by transmitting three fragments from nodes 1, 2 and 3. The repair-bandwidth of a DR storage node is smaller than the repair-bandwidth for a complete storage node.

\[
\{ (\alpha, \beta) | \beta(d - \theta) < \alpha \leq \beta(d - \theta + 1) \} \text{ for } \theta \in \{1, 2, \ldots, k - 2\}. \text{ Then the repairing storage capacity } \alpha_9 = (k - \theta)\beta.
\]

Note that in this setting for EMSR codes we have \((d + 1 - (k - 1)) \leq \alpha\), and for EMBR codes \(\alpha \leq (d + 1 - (0))\beta\), which corresponds respectively to \(\theta_{EMSR} = k - 1\), and \(\theta_{EMBR} = 0\). Thus in general \(\alpha_9 = (k - \theta)\beta\) is the optimal required repairing storage space for the extended regenerating codes.

Remark 4.1. We remark that the above conjecture is for functional repair and the optimal bound for exact repair is still not known.

4.5.6 Repair with DR Storage Nodes in Packet Erasure Networks

The above results show that repair bandwidth can be reduced if DR storage nodes are used for a distributed storage system with error-free channels. In what follows, we shall show that DR storage nodes can also reduce the repair-bandwidth in packet erasure networks. The following two corollaries illustrate more formally the effect of DR storage nodes in packet erasure networks. Since the analysis and code construction are similar to Section 4.4, we skip the proof.

**Corollary 4.1.** For a DSS with parameters \((n, k, d, h, \alpha = M/k, \alpha' = \beta, \gamma, M\xi, p)\) in a packet erasure network with channels having packet erasure probability \(p\),
4.6 Finite File Size Analysis

the asymptotic optimal repair-bandwidth for EMSR codes is $\gamma_{EMSR_{n,p}}/M = (d + h)/(k(d + h - k + 1)(1 - p))$.

Similarly, for EMBR codes, we have the following corollary.

**Corollary 4.2.** For a DSS with parameters $(n, k, d, h, \alpha, \alpha' = k\beta, \gamma, M\xi, p)$ in a packet erasure network with packet erasure probability $p$ for all channels, the asymptotic optimal repair-bandwidth for EMBR codes is $\gamma_{EMBR_{n,p}}/M = 2(d + h)/(k(2d + 2h - k + 1)(1 - p))$.

**Remark 4.2.** We note that a DR storage node behaves the same as a complete storage node in repair and transmits the same amount of repairing data as a complete storage node. That is, if $\beta'$ denotes the amount of data transmitted by a DR storage node in repair, $\beta' = \beta$. The analysis for $\beta' \neq \beta$ can be studied as future work, following a similar approach as the study of flexible regenerating codes in [SRK10], in which the surviving nodes are allowed to transmit different amount of repairing data.

4.6 Finite File Size Analysis

4.6.1 The Probability of Successful Repair

Above, we investigated the optimal repair-bandwidth in packet erasure networks under the assumption that the stored file contains an infinite number of packets (infinite order of subpacketization). In practice, a large order of subpacketization leads to high traffic load for packet headers and high complexity in decoding (e.g., by Gaussian elimination method, it requires the inverse of an $M\xi \times M\xi$ matrix). Meanwhile in some networks the packet size is fixed (or finite). Thus the number of packets in repair traffic might be finite. It is interesting to study the repair-bandwidth in the case of a finite order of subpacketization. Since the channel between the surviving nodes and the new node is lossy, then the repair may fail (the new node may not receive sufficient repair packets). We will first analyze the probability of successfully receiving $\beta$ fragments from a surviving node, denoted by $P_{\beta}$. Then, we will study the probability of successful repair (PSR). We note that the asymptotic optimal repair-bandwidth in Section 4.4 can serve as a lower-bound of the repair-bandwidth for a finite subpacketization order.

To successfully repair, the new node must receive $\beta$ fragments (that equals to $\beta\xi$ packets) from each of $d$ surviving nodes. Due to erasures, a surviving node transmits $t\xi$ packets produced by linear combinations (in GF($q$)) of $\beta\xi$ repair packets). For a packet erasure channel, $t = \beta/(1-p)$ if $\xi \rightarrow \infty$, which means infinite subpacketization order. For finite $\xi$, the total number of packet transmissions, $t$, may be much larger than $\beta/(1-p)$, and it depends on parameters $q$ and $\xi$. The successful receiving of $\beta$ fragments from a surviving node depends on two conditions. Firstly, $i$ packets must be received for $i \geq \beta\xi$. Secondly, $\beta\xi$ packets with independent encoding vectors among $i$ successfully received packets are available at the new
Figure 4.12: $P_\beta$ (numbers in the curves) in erasure networks for different values of $\xi$ and different values of bandwidth overhead ratio $t/\beta$. The field size here is $q = 5$ and $p = 0.3$. The larger $\xi$, the closer the bandwidth overhead ratio to the optimal value $1/(1-p) \approx 1.43$. As an example, the * symbol in the figure shows that for achieving $P_\beta \geq 0.92$, and for a given $\xi = 2$, each surviving node must transmit at least $1.77\beta$ fragments.

When $d$ surviving nodes are sending repairing packets, the probability of successful repair, denoted as $P_s$, is obtained by

$$P_s = P_\beta^d.$$  

4.6.2 Practical-Repair-Bandwidth

We formally define the practical-repair-bandwidth as the minimum required bandwidth to achieve $P_s \geq 1 - \delta$, where $\delta$ denotes as a parameter indicating how close the PSR is to 1 and $0 \leq \delta \leq 1$. Let $\hat{\gamma}(\delta, d)$ denote the practical-repair-bandwidth for given $\delta$ and $d$, then
4.6. Finite File Size Analysis

\[ \gamma(\delta, d) = \min \sum \delta t, \]
\[ \text{subject to: } P_s \geq 1 - \delta, \]  
(4.15)

where \( t \) is the number of packets transmitted by each surviving node.

4.6.3 Reducing Practical-Repair-Bandwidth

We have shown that DR storage nodes can reduce the repairing bandwidth for the scenario of infinite repair packets. Similarly, we can show that DR storage nodes reduce the bandwidth for finite repair packets. However, in what follows, we propose another method to reduce the practical repair-bandwidth. Consider a repair process in a distributed storage system. Packets on the links are erased i.i.d. with probability \( p \). Successful receiving \( \beta(d_1) \xi \) packets from \( d_1 \) surviving nodes guarantees successful repair. Suppose that the repair packets from each surviving node are independent of the set of \( d_1 \) nodes. This can happen for example by the use of product-matrix regenerating codes in [RSKR09b]. Then the new node by receiving from any set of \( d_1 \) nodes can successfully be generated. Thus, to increase the PSR, \( d_{tot} = d_1 + d_2, (d_2 \geq 0) \) surviving nodes transmit repairing packets. Each surviving node transmits \( t \xi \) packets, each of which is formed by a linear combination of \( \beta(d_1) \xi \) repair packets over \( GF(q) \). Hence, repair is successful if the new node receives \( \beta(d_1) \) fragments from at least \( d_1 \) links. We also note that the packets transmitted from one surviving node can only be used to decode the repairing information of that surviving node. Hence, the PSR is the probability by which the new node receives \( \beta(d_1) \) fragments from at least \( d_1 \) out of \( d_1 + d_2 \) surviving nodes. It is calculated by

\[ P_s = \sum_{i=d_1}^{d_1+d_2} \binom{d_1+d_2}{i} (P_\beta)^i (1-P_\beta)^{d_1+d_2-i}. \]  
(4.16)

The PSR for two schemes when \( d_1 = 9, d_2 = 0 \) and \( d_1 = 7, d_2 = 2 \) have been compared in Figure 4.13. For this example on both schemes MBR codes with parameters \( (n = 10, k = 5, M = 70) \) are used and other parameters are set as \( \xi = 5, q = 5 \) and \( p = 0.3 \). Moreover, the practical repair-bandwidth for \( P_s \geq 0.99 \) (i.e., \( \delta = 0.01 \)) has been compared between these two methods. We see that the scheme with smaller asymptotic optimal repair-bandwidth has almost twice larger practical-repair-bandwidth than the other scheme. This motivates the following optimization problem. Given the constraint that the PSR is greater than \( 1 - \delta \), we aim to minimize the practical-repair-bandwidth \( \tilde{\gamma}(\delta, d_1 + d_2) \) with variables \( d_1 \) and \( d_2 \). The

\footnote{\( \beta \) is a function of \( d_1 \) helper nodes}
Figure 4.13: Probability of successful repair in packet erasure networks for two schemes over the repair-traffic from each node. The first scheme aims at reducing asymptotic repair bandwidth and use \(d_1 = 9, d_2 = 0\). The second scheme uses \(d_1 = 7, d_2 = 2\). The figure (vertical lines) also compares practical repair-bandwidth regarding \(PSR \geq 0.99\) versus optimal asymptotic repair-bandwidth. Here, MBR codes are used and parameters are set as \(n = 10, k = 5, M = 70, \xi = 5, q = 5\).

Optimization problem can be formulated as follows,

\[
\begin{align*}
\min_{d_1, d_2} \quad & \widehat{\gamma}(\delta, d_1 + d_2) \\
\text{subject to:} \quad & P_s \geq 1 - \delta, \\
& d_1 + d_2 \leq d_{\text{tot}}.
\end{align*}
\]

It is not straightforward to find an analytical solution for the optimization problem. The optimal practical-repair-bandwidth solution depends on the probability of packet erasure on the links, the finite field size and the subpacketization order. We use the optimization problem in the previous example and find the corresponding \(d_1\) and \(d_2\) for the given finite field size \(q = 5\) and subpacketization order \(\xi = 5\) over the different packet erasure probabilities. \(d_1\) and \(d_2\) that minimize the repair-bandwidth are shown in Figure 4.14. We can conclude that for the network with higher erasure probabilities, more redundant data (larger \(d_2\)) should be used to reduce the practical repair-bandwidth. Conversely, for the network with lower erasure probabilities, less redundant data is needed and the optimal practical repair-bandwidth is closer to the optimal asymptotic repair-bandwidth.
4.7 Conclusions

We studied the repair problem for distributed storage systems with packet erasure channels and dedicated-for-repair (DR) storage nodes. We investigated the optimal storage-bandwidth tradeoffs for packet erasure networks. We proposed DR storage nodes to reduce the repair-bandwidth. A benefit of DR storage nodes is smaller storage capacity and repair-bandwidth, compared to complete storage nodes. We also studied the necessary minimal storage capacity for the DR storage nodes. We note that the benefit of DR storage nodes is not only limited to lossy networks, but they are also valid in reducing the repair-bandwidth in lossless networks. Then, we investigated the probability of successful repair and then proposed an approach to reduce the repair-bandwidth when the file size is finite. To reduce the practical repair-bandwidth, we formulated an optimization problem and showed that the optimal solution for that depends on the channel erasure probability.

Figure 4.14: The value of $d_1$ and $d_2$ that minimize the practical repair-bandwidth over different values of links packet erasure probability, $p$. Here, $\delta = 0.01$, and $p$ changes from 0.01 to 0.1. For a network with higher $p$ the reliability of repair becomes more critical and then repair-bandwidth is minimized when the redundant information is increased ($d_2$ increases). Here, $q = 5$, $\xi = 5$. 

1
4.8 Appendices

4.8.1 Proof of Lemma 1

Inspired by the analysis in [DGW+10] for distributed storage systems in lossless networks, we provide the proof for distributed storage systems with packet erasure channels and DR storage nodes. Another useful information-theoretic tool in the proof is the cut-set upper bound analysis introduced by Cover and Thomas [CT12].

Given a subset $V_x$ of nodes in a network (i.e., $V_x \subset \mathcal{N}$, where $\mathcal{N}$ is the set of nodes in the network), when all the nodes in $V_x$ perfectly cooperate and also all the nodes in $V^c_x$ perfectly cooperate, then the network can be modeled to a multiple-input multiple-output point to point erasure channel. Since the channels are independent and memoryless, the upper bound of the information transfer between $V_x$ and $V^c_x$ is when inputs to channels are independent and have uniform distribution among the set of alphabets. Thus the upper bound of information transfer is the sum of capacities of edges between $V_x$ and $V^c_x$.

We now show that the information rate in the modified information flow graph is upper bounded by $\sum_{i=0}^{k-1} \min\{\alpha \xi / \tau, (d + h - i)(1 - p)\beta \xi / \tau\}$. That is equal to show that the min-cut in the information flow graph equals to $\sum_{i=0}^{k-1} \min\{\alpha \xi / \tau, (d + h - i)(1 - p)\beta \xi / \tau\}$ [Ber98]. The information flow graph after $k$ failure/repair processes can be shown in $k$ subsequent stages, as depicted in Figure 4.1. We first prove there is a cut in the network with capacity $\sum_{i=0}^{k-1} \min\{\alpha \xi / \tau, (d + h - i)(1 - p)\beta \xi / \tau\}$. For the purpose, consider a cut that passes a route with a minimum capacity at any stage of repair. For example at stage 1, the cut selects a route between $\alpha \xi / \tau$ and $(d + h)(1 - p)\beta \xi / \tau$, where $(1 - p)\beta \xi / \tau$ is the capacity of the packet erasure link between one of the helper nodes and the new node. At stage 2, since the new node can get $(1 - p)\beta \xi / \tau$ packets per unit time from the previously generated node, then the cut selects between $\alpha \xi / \tau$ and $(d + h - 1)(1 - p)\beta \xi / \tau$, and so on. Finally, there will be a graph with a cut capacity equivalent to $\sum_{i=0}^{k-1} \min\{\alpha \xi / \tau, (d + h - i)(1 - p)\beta \xi / \tau\}$. Any other cut has a capacity larger than, or equal to, $\sum_{i=0}^{k-1} \min\{\alpha \xi / \tau, (d + h - i)(1 - p)\beta \xi / \tau\}$. This concludes that $\sum_{i=0}^{k-1} \min\{\alpha \xi / \tau, (d + h - i)(1 - p)\beta \xi / \tau\}$ is the min-cut, and consequently it is the upper bound of the information transfer rate between the source and a data-collector in this network.

4.8.2 Proof of Theorem 1

We use random linear codes to prove the achievability. The converse is proved as a corollary of Lemma 1, where we substitute $h = 0$ in the derived upper-bound.

Achievability: Let us first prove for the case that there is only one data-collector. Suppose packets $w_1, w_2, \ldots, w_M$ are the source data. There is a data collector which is connected to a subset of $k$ out nodes of complete storage nodes.

$V^c_x$ denotes the complement set of $V_x$. 
4.8. Appendices

We apply the random linear codes proposed in [LMKE08, LMKE05] for achieving the capacity of multicasting over lossy packet networks.

**Encoding:** Packets $v_1, v_2, \ldots, v_\Omega$ are constructed by random linear combinations of the original packets $w_1, w_2, \ldots, w_M$. That is, $v_l$ for $l = 1, \ldots, \Omega$ is

$$v_l = \sum_{i=1}^{M} \eta_{li} w_i,$$

where $\eta_{li}$ is selected randomly and uniformly from $GF(q)$. The packets $v_1, v_2, \ldots, v_\Omega$ are then injected to the source node according to a Poisson distribution by a constant rate $R_0 = (1 - \delta) \sum_{i=0}^{k-1} \min\{\alpha i/\tau, (d-i)(1-p)\beta i/\tau\}$, for some $\delta > 0$. For a node, in an occasion of a packet transmission, the packet is formed by a linear combinations of the packets that previously have been received by the node. Thus, a packet $x$, the input of channels, can be written as a linear combination of $v_1, \ldots, v_\Omega$, as

$$x = \sum_{j=1}^{\Omega} \pi_j v_j = \sum_{j=1}^{\Omega} \pi_j \sum_{i=1}^{M} \eta_{ji} w_i = \sum_{i=1}^{M} \left( \sum_{j=1}^{\Omega} \pi_j \eta_{ji} \right) w_i = \sum_{i=1}^{M} \phi_i w_i.$$  

Here, vector $\pi = [\pi_j]_{j=1}^{\Omega}$ is called the auxiliary encoding vectors associated with packet $x$ and vector $\phi = [\phi_i]_{i=1}^{M}$ is called the global encoding vectors associated with packet $x$. We say packet $x$ is innovative to node $i$, if the auxiliary encoding vector of packet $x$ is not in the span of the auxiliary encoding vectors of the previously received packets in node $i$. The global encoding vector of each packet is placed in the header of the packet.

**Decoding:** A data-collector by receiving $M\xi$ packets with independent global encoding vectors can successfully decode the source data.

**Side Information at Data Collector:** A data-collector for successfully decoding the file needs global encoding vectors of the received packets. The global encoding vector of each packet can be placed inside each packet’s header. The overhead of side information can be made negligible if the packet size is large enough.

**Analyzing the Probability of Error:** Using the above encoding and decoding, the error in decoding may happen due to two error events: the first event $(E_1)$ is that the decoder receives fewer than $M\xi$ innovative packets; second event $(E_2)$ is that decoder receives greater than or equal to $M\xi$ innovative packets, but there is no $M\xi$ independent packets (packets having independent global encoding vectors) such that $M\xi \times M\xi$ global encoding matrix become full rank.

The probability of the first error event can be derived by analyzing the propagation of innovative packets in the network. The results in [LMKE08] shows that if random network coding, as described above, is used to achieve the capacity
of multicast packet erasure networks, then the propagation of innovative packets in network follows the same as queuing network where each node in the network, works like a $M/M/1$ queuing system. That is, if $\phi_i$ is the arrival rate of innovative packets to node $i$, the node $i$ has service rate $\phi_e(1 - 1/q)/(1 - \delta)$. This queue to be stable it requires $\phi_i \leq \phi_e(1 - 1/q)/(1 - \delta)$. Thus we must have $q \geq 1/\delta$. When this queuing networks works for a long time, then the number of innovative packets in the network (denoted as $N_i$) is time-invariant random variable \cite{LMKE08}. Therefore, the data collector receives fewer than $M\xi$ innovative packets if $\Omega - N_i < M\xi$.

The probability that $N_i > \Omega - M\xi$ can be made arbitrarily small, by selecting $\Omega = \lceil M\xi/Rc \rceil$, for $Rc < 1$, and then selecting large $\xi$. (For more details, please refer to the proof of Theorem 1 in \cite{LMKE08}.)

Achievable Rate: Finally, by the above random network coding the rate is $M\xi/\tau$, where $M\xi$ is the total information transmitted and $\tau$ is the time taken for packets $v_1, \ldots, v_\Omega$ to reach to source node $S$ by rate $R_0 = (1 - \delta)\sum_{i=0}^{k-1}\min\{\alpha\xi/\tau, (d - i)(1 - p)\beta\xi/\tau\}$. For $\xi$ sufficiently large, $\Omega$ would be sufficiently large, then with a probability of error not exceeding $\varepsilon$ the rate would be,

\begin{equation}
\frac{M\xi}{\tau} > \frac{M\xi}{R(1/R_0 + \varepsilon)} \geq \frac{Rc R(1 - \Delta)}{1 + \varepsilon R(1 - \Delta)},
\end{equation}

which can be made arbitrarily close to $R = \sum_{i=0}^{k-1}\min\{\alpha\xi/\tau, (d - i)(1 - p)\beta\xi/\tau\}$. In the above equations, (4.20) holds because of Fano’s inequality, and (4.22) holds because $\Omega = \lceil M\xi/Rc \rceil$. By Lemma 1, it is impossible to achieve a rate greater than $R$. Thus, the rate $R = \sum_{i=0}^{k-1}\min\{\alpha\xi/\tau, (d - i)(1 - p)\beta\xi/\tau\}$ is the optimal achievable bound for this distributed storage system. Hence, for the optimal code we have $M\xi/\tau = \sum_{i=0}^{k-1}\min\{\alpha\xi/\tau, (d - i)(1 - p)\beta\xi/\tau\}$. Then, for finite but very large $\xi$ and $\tau$, and $\delta$ very close to zero, we define $C_0(\alpha) = \sum_{i=0}^{k-1}\min\{\alpha, (d - i)(1 - p)\beta\} = M$. Then, the optimal repair-bandwidth for varying individual node storage capacities
Appendices

can thus be derived by the similar approach adopted in [DGW+10]. Then,

\[
C_0(\alpha) = \begin{cases} 
    k\alpha & \alpha \in [0, b_0] \\
    b_0 + (k-1)\alpha & \alpha \in (b_0, b_1] \\
    \vdots \\
    b_0 + b_1 + \cdots + b_{k-1} & \alpha \in (b_{k-1}, \infty) 
\end{cases}
\] (4.28)

where,

\[
b_i \triangleq \left(1 - \frac{k-1-i}{d}\right) (1-p)\gamma.
\] (4.29)

Then, we find the minimum \(\alpha\), denoted as \(\alpha^* = C_0^{-1}(M)\). Hence,

\[
\alpha^* = \begin{cases} 
    \frac{M}{k} & M \in [0, kb_0] \\
    \frac{M - b_i}{(k-1)} & M \in (kb_0, b_0 + (k-1)b_1] \\
    \vdots \\
    M - \sum_{j=0}^{k-1} b_j & M \in \left(\sum_{j=0}^{k-2} b_j + b_{k-2}, \sum_{j=0}^{k-1} b_j\right)
\end{cases}
\] (4.30)

In general we have,

\[
\alpha^* = \frac{M - \sum_{j=0}^{i-1} b_j}{k-i},
\] (4.31)

where,

\[
M \in \left(\sum_{j=0}^{i-1} b_j + (k-i)b_{i-1}, \sum_{j=0}^{i} b_j + (k-i-1)b_i\right).
\] (4.32)

Now from the definition of \(b_i\), we can verify that,

\[
\sum_{j=0}^{i-1} b_j = (1-p)\gamma g(i),
\] (4.33)

and

\[
\sum_{j=0}^{i} b_j + (k-i-1)b_i = \gamma \frac{M}{h(i)}.
\] (4.34)

Thus, we have,

\[
\alpha^* = \frac{M - (1-p)\gamma g(i)}{k-i},
\] (4.35)

where,

\[
M \in \left(\frac{\gamma M}{h(i-1)}, \frac{\gamma M}{h(i)}\right).
\] (4.36)

By deriving \(\gamma\) based on \(M\), the result as in [DGW+10] is obtained.
4.8.3 Proof of Lemma 4.2

The proof is based on using random linear codes for storing the original file in storage nodes and in the repair process. Let \( Q_i \) denote the encoding matrix of a complete storage node \( i \) such that each row of the \( Q_i \) represents the encoding vector of a fragment stored in node \( i \). Similarly, let \( Q_h \) denote the encoding matrix of DR storage node \( j \). Each row of the \( Q_h \) represents the encoding vector of a fragment stored in DR storage node \( j \). By exploiting sparse-zero lemma \cite{Yeung08}, we show for a large finite field size there exist codes \( Q_i \) for \( i = 1, \ldots, n \) and \( Q_h \) for \( j = 1, \ldots, h \) such that the new node is generated by linear codes and the repair-bandwidth is optimal.

For the code construction, we split the source file of a size \( M \) into \( k(d + h - k + 1) \) fragments. We denote the source file by vector \( x = [x_1, x_2, \ldots, x_{k(d + h - k + 1)}]^T \).

Substituting these set of parameters \((d, h, M = k(d + h - k + 1))\) in Eq. (4.10), we have \( \beta = 1 \). We construct an \((n, k, d)\) - EMSR\( h \) code (with a little abuse of notation \((n, k, d)\) - EMSR\( p=0, h \) codes) requiring the minimum repair-bandwidth. Before that, we define a property, denoted as the extended MDS property, as follows.

**Definition 4.5 (Extended MDS Property).** For a set of \( n \) complete storage nodes and \( h \) DR storage nodes, we say it has the extended MDS property if: I) for any subset of \( k \) complete storage nodes the matrix of their encoding vectors has full rank; II) for any set of one complete storage node plus \( h \) DR storage nodes the matrix of their encoding vectors has full rank; III) for any set of \( k - 1 \) complete storage nodes plus \( h \) DR storage nodes the matrix of their encoding vectors has full rank.

Finding a code with conditions I and II is not difficult. However, finding a code satisfying Condition III may look challenging. We note that in our setting, we can find such a code because we always have \( h = h\beta < M - (k - 1)\alpha = d + h - k + 1 \). That is, there are always some vectors orthogonal to the span of encoding vectors in the set of \( k - 1 \) complete storage nodes plus \( h \) DR storage nodes (because \( h\beta + (k - 1)\alpha < M \)).

**Lemma 4.5** (Sparse-Zero Lemma). Consider a multi-variable polynomial \( g(\alpha_1, \alpha_2, \ldots, \alpha_n) \) which is not identically zero, and has the maximum degree in each variable at most \( d_0 \). Then, there exist variables \( \gamma_1, \gamma_2, \ldots, \gamma_n \) in the finite field \( GF(q) \), for \( q \geq d_0 \), such that \( g(\gamma_1, \gamma_2, \ldots, \gamma_n) \neq 0 \).

**Proof.** See proof of Lemma 19.17 in \cite{Yeung08}.

**Lemma 4.6.** Consider an \((n, k, d)\) - EMSR\( h \) code satisfying the Extended MDS property. For any set of selecting \( k - 1 \) complete storage nodes, it is possible to select one vector from each of \( h \) DR storage nodes and one vector from \( d - k + 1 \) nodes such that resulting matrix of \((k - 1)(d + h - k + 1) + h + (d - k + 1) = k(d + h - k + 1)\) encoding vectors has full rank.
Proof. For an encoding matrix $Q_i$, let $\text{span}(Q_i)$ denotes the span of their row vectors. For encoding vectors of a set of $k - 1$ nodes, let say $Q_{s_1}, \ldots, Q_{s_{k-1}}$ and $h$ DR storage nodes, we prove there exists an encoding vector in $Q_{s_i}$ that is not in $\text{span}(Q_{s_1}, \ldots, Q_{s_{k-1}}, Q_{h_1}, \ldots, Q_{h_h})$. The proof is based on a contradiction argument.

If there is not an encoding vector in $Q_{s_i}$, for $i \notin \{1, \ldots, k-1\}$, then the code cannot have the Extended MDS property. This is because, by code construction $Q_{s_i}$ is not in the span of $Q_{h_1}, \ldots, Q_{h_h}$. Therefore, if there exists no encoding vector in $Q_{s_i}$ that is out of $\text{span}(Q_{s_1}, \ldots, Q_{s_{k-1}}, Q_{h_1}, \ldots, Q_{h_h})$, then $Q_{s_i}$ is in the span of $Q_{s_1}, \ldots, Q_{s_{k-1}}$. This means $Q_{s_1}, \ldots, Q_{s_{k-1}}, Q_{s_k}$ is not full rank. This is a contradiction to the Extended MDS property.

Theorem 4.4. Consider a distributed storage system storing a file by an $(n, k, d)$-EMSR$_h$ code and having Extended MDS property. For any finite field $GF(q)$, with

$$q \geq d_0 = (d + h - k + 1)k \left(\binom{n}{k} + 2\binom{n-1}{k-1} + 2n\right),$$

(4.37)

there exists a linear code for storing a file in the system such that it maintains the Extended MDS property before/after repair in which it requires the minimum repair-bandwidth.

Proof. The proof is by induction. We construct a code that has the Extended MDS property and maintains that property after repair. To this aim, we first find coefficients in $n(d + h - k + 1)$ encoding vectors of $Q_1, \ldots, Q_n$ (codes in complete storage nodes), and coefficients in $h$ encoding vectors of $Q_{h_1}, \ldots, Q_{h_h}$, codes in DR storage nodes such that:

$$\prod_{\{s_1, \ldots, s_k\} \subseteq 1, \ldots, n} \text{det}[Q_{s_1}, \ldots, Q_{s_k}] \times$$

$$\prod_{\{s_i\} \subseteq 1, \ldots, n} \text{det} \left[ Q_{s_1}, Q_{h_1}, \ldots, Q_{h_h} \right] A_{1}^T \right] \times$$

$$\prod_{\{s_1, \ldots, s_{k-1}\} \subseteq 1, \ldots, n} \text{det} \left[ Q_{s_1}, \ldots, Q_{s_{k-1}}, Q_{h_1}, \ldots, Q_{h_h} \right] A_{2}^T \right] \neq 0.$$  

(4.38)

As $h\beta + (k - 1)\alpha < M$, then there exist some assignments for the coefficients which makes the left-hand side in (4.38) as a non-zero value. Thus we can use sparse-zero Lemma. The maximum degree of each coefficient is not greater than $d_0$. This implies that there exist some coefficients in $GF(q)$, for $q > d_0$, which make the left-hand side of (4.38) a non-zero value.
Now, suppose that we have a code with the Extended MDS property in the storage nodes. We show that the repair is possible with the minimum repair-bandwidth if $q > d_0$. Let $Q'_1$ be the code on the new nodes which is generated by

$$Q'_1 = [Q_2 b_2, \ldots, Q_{d-k+1} b_{d-k+1}, Q_{h_1} b_{h_1}, \ldots, Q_{h_h} b_{h_h}] Z,$$  \hspace{1cm} (4.39)

where $\{b_2, \ldots, b_{d-k+1}\}$, $\{b_{h_1}, \ldots, b_{h_h}\}$, and $Z$ are proper vectors with arrays (coefficients) that we can freely select such that the new code has the Extended MDS property. That is we select these free coefficients such that

$$\prod_{\{s_1, \ldots, s_{k-1}\} \subseteq 2, \ldots, n} \det [Q'_1, Q_{s_1}, \ldots, Q_{s_{k-1}}] \times$$

$$\det \left( \left[ Q'_1, Q_{h_1}, \ldots, Q_{h_h} \right] [A_3]^T \right) \times$$

$$\prod_{\{s_1, \ldots, s_{k-1}\} \subseteq 2, \ldots, n} \det \left( \left[ Q'_1, Q_{s_1}, \ldots, Q_{s_{k-2}}, Q_{h_1}, \ldots, Q_{h_h} \right] [A_4]^T \right) \neq 0.$$  \hspace{1cm} (4.40)

Using Lemma 4.6 there will be some assignment for coefficients which make the term in the left-hand side of the above equation a non-zero value. Hence the left-hand side of the above equation is a non-zero polynomial. Therefore, we can use sparse-zero Lemma. Again, since the maximum degree of each free coefficient is

$$(d + h - k + 1) k \left( \frac{n - 1}{k - 1} \right) + 2 \left( \frac{n - 2}{k - 2} \right) + 2,$$  \hspace{1cm} (4.41)

which is smaller than $d_0$, then there exist some coefficients in $GF(q)$, for $q > d_0$, which make the left-hand side of (4.40) as a non-zero value. \hfill \Box

4.8.4 **Proof of Lemma 4.3**

We prove for $h = 1$. For other values of $h$, the proof follows the same. For an EMBR$_{h=1}$ code, we have $\sum_{i=0}^{k-1} (d + 1 - i) \beta = M$. We now prove by a contradiction argument. Suppose $\alpha' < k \beta$. Let us first consider a set of $\alpha'$ that $(k-1)\beta \leq \alpha' < k \beta$. Let $S$ denote the random variable representing the source file. Suppose $S$ takes a value randomly and uniformly taken from a set $S = \{1, \ldots, 2^M\}$. That is the source file contains $M$ bits of information, and we have $H(S) = M$, where $H(X)$ refers to the binary entropy of the random variable $X$. Next, let $W_l$ denote the random variable representing the content of node $l$ for $l \in [n]$. Assume $W_{n+i}$

$^4H(X) = \mathbb{E}\{-\log_2 P(X)\}.$
denotes the corresponding random variable for the content of the new node in i-th stage of repair. Since every k nodes have to reconstruct the original file, we have,

$$M = H(W_{n+1}, W_{n+2}, \cdots, W_{n+k})$$  \hspace{1cm} (4.42)

$$= H(W_{n+1}) + H(W_{n+2} | W_{n+1}) + \cdots$$

fr. CS s fr. DR fr. CS s fr. DR

$$+ H(W_{n+k} | W_{n+1}, \cdots, W_{n+k-1})$$  \hspace{1cm} (4.43)

$$= \underbrace{d\beta + \beta \cdots} \hspace{0.5cm} \text{(stage 1)}$$

fr. CS s fr. DR

$$+ \underbrace{(d - (k - 2))\beta + \beta \cdots} \hspace{0.5cm} \text{(stage 2)}$$

fr. CS s fr. DR

$$+ \underbrace{(d - (k - 1))\beta + (\alpha' - (k - 1)\beta)} \hspace{0.5cm} \text{(stage k)}$$  \hspace{1cm} (4.44)

$$= (d + 1)\beta + (d)\beta + \cdots$$

$$+ (d + 1 - (k - 1))\beta + (\alpha' - k\beta)$$  \hspace{1cm} (4.45)

$$= \sum_{i=0}^{k-1} (d + 1 - i)\beta + (\alpha' - k\beta),$$  \hspace{1cm} (4.46)

$$= M + (\alpha' - k\beta),$$  \hspace{1cm} (4.47)

$$< M.$$  \hspace{1cm} (4.48)

which is a contradiction.

In the above equations, CSs stands for the complete storage nodes and DR stands for the DR storage node. In the proof, (4.42) follows by the chain rule of entropy, and (4.43) follows from the fact that each new node in stage i, conditioning on knowing information in previous stages, receives at most \( (d+1-i)\beta \) information. Also (4.44) follows from the fact that at stage k, the DR storage node can send at most \( \alpha' - (k-1)\beta \) (new) information to node \( n+k \). Also (4.45) follows from the fact that \( \sum_{i=0}^{k-1} (d + 1 - i)\beta \) = M. Finally, (4.48) follows from \( \alpha' - k\beta < 0 \). The same argument can be followed for \( \alpha < (k-1)\beta \). This finalizes the proof.

4.8.5 Proof of Lemma 4.3

We propose linear codes for achieving the bound \( \sum_{i=0}^{k-1} (d + h - i)\beta = M \) where there are h DR storage nodes and d complete storage nodes in the repair process. Our proof is inspired by the argument adopted in [WDR07]. For the code construction we first define a test, denoted as the extended rank test.
Definition 4.6 (Extended Rank Test). We say that a linear code $Q_1, \ldots, Q_s, Q_h, \ldots, Q_{k'}, Q_{k'+1}, \ldots, Q_n$ in $GF(q)$ passes the extended rank test by a test vector $[h_1, \ldots, h_k, h_{h_1}, \ldots, h_{h_{h_1}}, h_{k+1}, \ldots, h_n]$, for $0 \leq h_i \leq \alpha$ and $0 \leq h_{h_i} \leq k\beta$, if matrix $[Q_1E_1, \ldots, Q_sE_k, Q_hE_{h_1}, \ldots, Q_{k'}E_{k'}, Q_{k'+1}E_{k'+1} \ldots Q_nE_n]$ has full rank $M$, where $Q_iE_i$ denotes the first $h_i$ vectors of $Q_i$.

We may also say that a code passes an extended rank test weakly by test vector $h$, if matrix $[Q_1E_1, \ldots, Q_sE_k, Q_hE_{h_1}, \ldots, Q_{k'}E_{k'}, Q_{k'+1}E_{k'+1} \ldots Q_nE_n]$ has full rank $M$, conditioning that we allow in each node to linearly combine the code vectors in $Q_i$ and send $h_i$ coded vectors. A test vector has $1 \times (n + h)$ dimension based on the following pattern:

$$h = (h_1, \ldots, h_k, h_{h_1}, \ldots, h_{h_{h_1}}, h_{k+1}, \ldots, h_n).$$

(4.50)

Note that the positions $k + 1, k + 2, \ldots, k + h$ in the extended rank test vector are dedicated for DR storage nodes. Now our objective is to design a code such that the code maintains the property of passing extended rank test over the following set of vectors:

$$H_0 \triangleq \{h | h \text{ is a permutation of } h^{(0)}\},$$

(4.51)

where

$$h^{(0)} = [\alpha, \ldots, \alpha, 0, \ldots, 0].$$

(4.52)

Here, the permutation of a $n + h$-dimensional vector $h$ is defined in this way that the arrays whose positions in the vector belong to complete storage nodes can permute between themselves, and arrays whose positions in the vector belong to DR storage nodes can permute between themselves. For constructing $(n, k, d) - \text{EMBR}_h$ codes, we find a set $H \supset H_0$, such that if the code before any node failure (let say old code) passes the extended rank test by vectors in $H$, the code after repair (let say new code) can also pass the test. Suppose such $H$ is given, by the following theorem we states that it is always possible to construct a linear $(n, k, d) - \text{EMBR}_h$ code over sufficiently large finite field size. More formally, we state in the following theorem.

Theorem 4.5. Suppose a set of test vectors $H$ with the following properties is given:

- $H \supset H_0$;
- $\forall h \in H$, where $h = [h_1, \ldots, h_k, h_{h_1}, \ldots, h_{h_{h_1}}, h_{k+1}, \ldots, h_n]$, we have $0 \leq h_i \leq \alpha$ and $0 \leq h_{h_i} \leq k\beta$;
- $\forall h \in H$, $w(h) \triangleq \sum_{i=1}^{n+h} h_i \geq M$;
- For any $h \in H$ and any permutation of $1, \ldots, n$, let say $s_1, \ldots, s_n$, there exists a vector $h' \in H$ that
Then there exists a linear code for constructing an \((n, k, d)−EMBR_h\) code, provided that the finite field size \(q\) is greater than
\[
q \geq \max \left\{ \left( \frac{n\alpha + h k \beta}{M} \right), 3|\mathcal{H}|M \right\}. \tag{4.57}
\]

Proof. The proof is again based on induction. The proof is inspired by the proof of Theorem 3 in [WDR07]. We remark that the set \(\mathcal{H}\) for EMBR code is different from the analysis in [WDR07]. We maintain the property that
\[
\prod_{h \in \mathcal{H}} \det \left( \begin{bmatrix} Q_1 E_{h_1} & \ldots & Q_n E_{h_n} & Q_{h_1} E_{h_1} & \ldots & Q_{h_{h_n}} E_{h_{h_n}} \end{bmatrix} \right) \neq 0. \tag{4.58}
\]

Let us initialize the system by constructing a code which passes the extended rank test by the set \(\mathcal{H}\). For that, we first construct a matrix \(G\), as a generator matrix of an \((n\alpha + h k \beta, M)\) MDS code. For this construction the finite field size \(q > \left( \frac{n\alpha + h k \beta}{M} \right)\) is sufficient. From the MDS code property, we know that every set containing \(M\) or greater than \(M\), row vectors of \(G\) has rank \(M\). Now, if we assign the row vectors as the code vectors of complete storage nodes and \(h\) DR storage nodes, we see that the constructed code passes the extended rank test.

Now, suppose that node 1 fails and we want to generate a new node such that the new code also passes the extended rank test. Suppose the code on the new node is denoted as \(Q_1'\), then
\[
Q_1' = [Q_{s_2} B_{s_2}, \ldots, Q_{s_{d+1}} B_{s_{d+1}}, Q_{s_1} B_{s_1}, \ldots, Q_{s_{h_n}} B_{s_{h_n}}]Z. \tag{4.59}
\]

Now, let the new node download \(h'_{s_i} - h_{s_i}\) from the complete storage node \(i\), for \(i \in \{2, \ldots, d+1\}\) and also download \(h'_{s_{h_i}} - h_{s_{h_i}}\) from the DR storage node \(i\), for \(i \in \{1, \ldots, h\}\). Since \(\alpha \geq h_{s_1} \geq \sum_{i=2}^{d+1} h'_{s_i} - h_{s_i} + \sum_{i=1}^{h} h'_{s_{h_i}} - h_{s_{h_i}},\) the new node can store this data in its nodes such that resulted code passes the rank test by the vectors in set \(\mathcal{H}\). Therefore, there exist an assignment such that
\[
\det \left( \begin{bmatrix} Q_1 E_{h_1} & \ldots & Q_n E_{h_n} & Q_{h_1} E_{h_1} & \ldots & Q_{h_{h_n}} E_{h_{h_n}} \end{bmatrix} \right) \neq 0. \tag{4.60}
\]

Thus, the polynomial is non-zero polynomial and we can use sparse-zero Lemma. We can verify that the maximum degree of a coefficient is not greater than \(3|\mathcal{H}|M\). Hence, there exist coefficients in \(GF(q)\) that make all the products non-zero.
Now, we describe how to construct the set $\mathcal{H}$ given in Theorem 4.5. We construct for $d = n - 1$. For other values of $d$, it needs further investigation. We construct $\mathcal{H}$ containing vectors $h^{(j)}$, for $j = 0, 1, \ldots, k$ and their permutations, where

$$h^{(j)} \triangleq (\alpha, \alpha, \ldots, \alpha, j^{(j)}\beta, j^{(j)}\beta, \ldots, j^{(j)}\beta, (j-1)\beta, \ldots, 2\beta, \beta, 0).$$

(4.61)

Next theorem states that the above construction satisfy the requirements for the test set $\mathcal{H}$.

**Theorem 4.6.** The set of test vectors constructed by the above method satisfies the following conditions:

- $\mathcal{H} \supseteq \mathcal{H}_0$;
- $\forall h \in \mathcal{H}$, where $h = [h_1, h_2, \ldots, h_k, h_{k+1}, \ldots, h_{n}]$, we have $0 \leq h_i \leq \alpha$ and $0 \leq h_{k+i} \leq k\beta$;
- $\forall h \in \mathcal{H}$, $w(h) \triangleq \sum_{i=1}^{n+h} h_i \geq M$;
- For $j = 0, \ldots, k$, $h^{(j)}$ consists of $(k-j)\alpha$’s followed by a $\beta$-gradually-decreasing vector ending in 0. Here, a vector $(x_1, x_2, \ldots, x_n)$ is $\beta$-gradually-decreasing vector if $x_1 \geq x_2 \geq \cdots \geq x_n$, and the difference between two consecutive elements is not larger than $\beta$.
- For any $h \in \mathcal{H}$ and any permutation of $1, \ldots, n$, let say $s_1, \ldots, s_n$, there exists a vector $h' \in \mathcal{H}$ that

$$h'_{s_1} = 0$$

(4.62)

$$h_{s_1} \geq \sum_{i=2}^{d+h+1} h_i$$

(4.63)

$$h'_{s_1} \in [h_{s_1}, h_{s_1} + \beta], \text{ for } i = 2, \ldots, d + h + 1,$$

(4.64)

$$h'_{s_1} = h_{s_1} \text{ for } i = d + h + 2, \ldots, n + h.$$  

(4.65)

It is straightforward to verify that $\mathcal{H}$ satisfies Properties 1, 2, 3, and 4. We next provide the proof for Property 5. Note that for the EMBR code, we have test vectors $h^{(j)}$’s and their permutations, for $j \in \{1, \ldots, k\}$. Also note that in our construction positions $k+1, \ldots, k+h+1$ are dedicated to DR storage nodes and thus $s_1 \notin \{k+1, \ldots, k+h+1\}$. For the proof, we study different cases, as follows.

**Case 1:** If $j = k$, or $s_1 > k - j$, to satisfy Property 5, we choose $h'$ as the following vector:

$$h' = (h_1, \ldots, h_{s_1-1}, h_n = 0, h_{s_1}, \ldots, h_{n-1}),$$

(4.66)

which is a permutation of $h$ and hence is in $\mathcal{H}$. 
4.8. Appendices

Case 2: If \( j \in 0, \ldots, k - 1 \), and \( s_1 \leq k - j \), then

\[
h^{(j)} = (\alpha, \ldots, \alpha, j\beta, j\beta, \ldots, j\beta, (j+1)\beta, \ldots, 2\beta, \beta, 0)).
\]

(4.67)

To satisfy Property 5, we choose \( h' \) as the following vector:

\[
h' = (\alpha, \ldots, \alpha, 0, \alpha, \ldots, \alpha, (j+1)\beta, \ldots, (j+1)\beta, j\beta, \ldots, j\beta),
\]

(4.68)

which is a permutation of \( h^{(j+1)} \) and hence is in \( \mathcal{H} \).

4.8.6 Proof of Proposition 4.1

For EMSR codes: For simplicity, and without loss of generality [?], assume \( \beta = 1 \). By Theorem 2, we know \( \alpha' = \beta = 1 \). We show that there is a linear combination of content of the \( k \) complete storage node that makes the content of the new DR storage nodes. Assume the global encoding vector of the fragment of the new node is \( \nu_1 \). Assume the global encoding vectors of \( M/k \) fragments of the complete storage node \( i \) is \( [\omega_{ij}]_{j=1}^{M/k} \), in which each column represents the global encoding vector of a fragment. Then, we show that there is a vector \([\gamma_1 \ldots \gamma_M]\) that

\[
[\gamma_1 \ldots \gamma_M] \begin{pmatrix} [\omega_{1j}]_{j=1}^{M/k} & [\omega_{2j}]_{j=1}^{M/k} & \ldots & [\omega_{kj}]_{j=1}^{M/k} \end{pmatrix}^T = \nu_1.
\]

(4.69)

Since every \( k \) complete storage nodes can reconstruct the original file, then matrix \( A \) is invertible. Therefore, there is a solution to the problem. After finding \([\gamma_1 \ldots \gamma_M]\), each node sends a fragment which is a linear combination of its stored fragment. For example node 1 sends \([\gamma_1 \ldots \gamma_{M/k}]\omega_{1j}]_{j=1}^{M/k} T \) and so on. Then, the new DR storage node generate its content by

\[
\nu_1 = \underbrace{[\gamma_1 \ldots \gamma_{M/k}]\omega_{1j}]_{j=1}^{M/k} T}_{\text{from node 1}} + \ldots + \underbrace{[\gamma_{M(k-1)/k+1} \ldots \gamma_M]\omega_{kj}]_{j=1}^{M/k} T}_{\text{from node } k}.
\]

(4.70)

Therefore, \( k\beta \) fragments are sufficient to generate the new DR storage node.

For EMBR codes: Again, assume \( \beta = 1 \). By Theorem 3, we know \( \alpha' = k\beta = k \). Since the DR storage node could help regenerating a complete storage nodes by sending \( \beta = 1 \) fragments and since for EMBR codes all the information received in repair by a complete storage node is saved in the storage (i.e., \((d + h)\beta = \alpha\)) then we can conclude that each DR storage node and a complete storage node have \( \beta \).
fragments of information in common. Thus, in repairing a DR storage node, \( k \) complete storage nodes equally send \( k\beta \) fragments to the new DR storage node. More formally, assume \( \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k \) are the encoding vectors of \( k \) stored fragments in the (failed) DR storage node. The DR storage node in repair of node \( i_1, i_2, \ldots, i_k \) could send respectively \( \mathbf{a}_1\mathbf{x}_1, \mathbf{a}_2\mathbf{x}_2, \ldots, \mathbf{a}_k\mathbf{x}_k \). Since the new node stores all the fragments that it receives, then \( \mathbf{a}_1\mathbf{x}_1 \) is available in node \( i_1 \), \( \mathbf{a}_2\mathbf{x}_2 \) is available in node \( i_2 \), and so on. In repairing a DR storage nodes, \( i_1, i_2, \ldots, i_k \) send \( \mathbf{a}_1\mathbf{x}_1, \mathbf{a}_2\mathbf{x}_2, \ldots, \mathbf{a}_k\mathbf{x}_k \). Since \( \mathbf{a}_i \)'s are independent vectors (if not, the data-collector could not reconstruct the original file by connecting to those \( k \) complete storage nodes) then the new DR storage node can recover lost fragments \( \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k \) by Gaussian elimination method. Thus, again \( k\beta \) fragments are sufficient.

### 4.8.7 Probability of Successful Repair by Random Linear Codes

Suppose that a surviving node is transmitting a number of packets toward the new node. Each packet in our model is a random linear combination of \( \beta\xi \) independent packets (having independent encoding vectors). Consider a case when the new node receives \( i \) packets from the surviving node. Let \( D_i \) be the random variable denoting the dimension of subspace spanned by all \( i \) vectors. Then, by random matrix arguments [ADMK05], for \( i \geq d \), we have

\[
\Pr(D_i = d) = \frac{\prod_{j=0}^{d-1}(q^j - q^1) \prod_{j=0}^{d-1}(q^{\beta\xi} - q^j)}{q^{d\beta\xi} \prod_{j=0}^{d-1}(q^d - q^j)}.
\]

(4.71)

Hence, the probability of having \( d = \beta\xi \) dimensions by receiving \( i \geq \beta\xi \) vectors equals to

\[
\Pr(D_i = \beta\xi) = \frac{\prod_{r=0}^{d-1}(q^r - q^\beta\xi)}{q^{d\beta\xi}}.
\]

(4.72)

That is the probability of having \( \beta\xi \) independent packets by receiving \( i \geq \beta\xi \) packets. Since packets are lost in the channel independently and with probability \( p \), then the probability of receiving \( i \) packets from \( t\xi \) transmitted packets has a binomial distribution and can be calculated as

\[
\binom{t\xi}{i}(1-p)^i p^{(t\xi-i)}.
\]

(4.73)

Combining this result with (4.72) gives us the probability of having \( \beta\xi \) independent packets by transmitting \( t\xi \) packets as

\[
P_\beta = \sum_{i=\beta\xi}^{t\xi} \binom{t\xi}{i}(1-p)^i p^{(t\xi-i)} \frac{\prod_{r=0}^{\beta\xi-1}(q^r - q^\beta\xi)}{q^{\beta\xi i}} \text{ if } t \geq \beta.
\]

(4.74)

Thus, the formula in Section 4.6.1 is resulted.
Part III

Partial Node Failure/Repair in Distributed Storage Systems
In this chapter, we consider a distributed storage system where parts of the stored packets in storage nodes are subject to being lost. In a process, termed as the partial repair, the lost packets in a faulty node are recovered by the transmitted packets from other storage nodes and the available packets in the faulty node. To improve reliability of the stored data, and reduce the transmission costs, we propose two-layer coding for storing files in the system. We study the minimum partial-repair bandwidth, and the codes that achieve the optimal bound.

5.1 Introduction

Coding in distributed storage systems has attracted significant research interests recently [DGW+10, W10b, NA15], especially the role in the repair process, where a new node is generated if a node fails. In [DGW+10] the optimal storage-bandwidth tradeoff is derived, and in [DGW+10, W10b] the erasure codes that achieve the tradeoff are proposed. In the existing works, generally the node failure is considered in the repair problem, where all the data in a storage node is lost.

Storage at each computing system (machine, or a set of collocated machines, termed here as nodes) can suffer from partial failure. For example, for a single disk, a partition could fail [BHHL13]; or for a file server a subset of its disks could fail. In data centers, where storage nodes are connected through a hierarchical network structure [AFLV08], storage nodes inside a rack communicate through the rack server. Other storage nodes in other racks are connected by higher layer servers. By neglecting the transmission costs inside a rack, partial failure can also occur in data centers. We term a storage node that has lost a part or all of its stored packets as a faulty storage node and a storage node without any lost packet as a correct storage node. To maintain the reliability of the stored file in the system, in a process,

\footnote{A packet contains a fixed number of bits, and is considered as one unit of information.}
termed as the partial repair, a number of correct storage nodes transmit sufficient numbers of packets to the faulty storage node. Note that the faulty node still has access to its non-erased packets as side information, which can be useful in the partial-repair process. Availability of side information has provided the benefit in reducing the transmission cost in the previous literature, e.g., in [MAN14, GXS15]. Yet, the storage-bandwidth in partial-repair has not been studied. We shall derive the optimal storage-bandwidth tradeoff in partial repair. Moreover, we will suggest two-layer source coding to achieve the optimal bound.

The organization of the chapter is as follows. In Section 5.2, we will study the fundamental storage-bandwidth tradeoff in partial repair. We will provide the proof of our main result in Section 5.3. Finally, we shall conclude the chapter in Section 5.4.

5.2 Storage-Bandwidth Tradeoff in Partial Repair

In this section, we will derive the optimal bandwidth-storage tradeoff in partial-repair. We assume two-layer source coding is used for storing a file in storage nodes. That is, in the first layer (outer layer), the source file of size $M$ is encoded by an $(n, k)$-erasure code and distributed to $n$ storage nodes such that every node stores $\alpha_i \geq M/k$ and every set of $k$ nodes can reconstruct the original file. In the second layer (inner layer), in each storage node $\alpha_i$ packets are encoded to $\alpha_o$ packets such that every set of $\alpha_i$ packets can recover $\alpha_i$ coded packets of the first layer. If we denote $\delta$ as the inner-layer code rate, then the total number of packets in a node is $\alpha_o = \alpha_i (1 + \delta)$. Next, assume that a node is faulty, and it lost $\xi \alpha_i$ packets. Here, we study for $\delta \leq \xi \leq 1 + \delta$. If $\xi \leq \delta$, there is a trivial solution by recovering the lost fragments in the faulty node, without any transmission between storage nodes. If a node completely fails, then $\xi = 1 + \delta$. In partial-repair, $d$ correct storage nodes equally transmit $\beta$ packets to the under-recovery storage node. To study the required bandwidth, we use a graph to represent the information flow in partial repair.

The information flow in partial repair is represented by a directed acyclic graph $G(n, k, d, \alpha_i, \delta, \beta, \xi)$, which consists of five types of nodes: source nodes, correct storage nodes, faulty storage nodes, under-recovery storage nodes and data collectors (DC). The source node distributes the original file among storage nodes by (presumably) infinite-capacity links. Every storage node is denoted by an input (in) and an output (out) node connected by a link of capacity $\alpha_i$. Note that the amount of stored information in a storage node is $\alpha_i$ packets, and $\delta \alpha_i$ packets are the redundant data in each storage node. To regenerate the lost packets, $d$ correct nodes sent packets to the under-recovery storage node. A directed edge from the out node of a correct storage node to the in node of the recovered node with capacity $\beta$ represents the amount of information sent from each of $d$ correct storage nodes. A directed edge from the out node of the faulty node to the in node of the under-recovery node with capacity $\alpha_i(1 - \xi + \delta)$ represents the amount of side information...
5.2. Storage-Bandwidth Tradeoff in Partial Repair

available at the under-recovery node. Finally, a DC reconstructs the stored file by connecting to \( k \) storage nodes via infinite-capacity edges and then solving a set of linear-equations using, e.g., the Gaussian elimination method. An interesting property in the information flow graph is that it maps the partial repair problem into the information flow problem in multicast networks. The well-known cut analysis in such networks [ACLY00, Yeu08] thus can be used here in characterizing the optimal partial repair bandwidth. An information flow graph for one stage of partial repair on node 4 in a four-node storage network has been shown in Fig. 5.2. In this example, a DC can reconstruct the file of size \( M \) if \( \beta \geq (M - (2 - \xi + \delta)\alpha_i)/2 \), by cut-set bound analysis.

In a more general setting than the above example, we derive the feasible region for partial repair and then characterize the tradeoff between storage and partial repairing bandwidth, as stated in the following theorem.

**Theorem 5.1.** Consider a distributed storage system with parameters \((n, k, d, \alpha_i, \delta, \beta, \xi, M)\). For given \( \delta, \gamma = d\beta \), and \( \xi \), the point corresponding to parameters \((n, k, d, \alpha_i, \delta, \beta, \xi, M)\) is feasible if and only if

\[
\sum_{i=0}^{k-1} \min \left[ \alpha_i, (1 - \frac{i}{d})\gamma + \alpha_i(1 - \xi + \delta) \right] \geq M, \quad (5.1)
\]

or equivalently

\[
\sum_{i=0}^{k-1} \min \left[ \alpha_i(\xi - \delta), (1 - \frac{i}{d})\gamma \right] + k\alpha_i(1 - \xi + \delta) \geq M. \quad (5.2)
\]
Figure 5.2: Information flow graph in partial repair, for \( n = 4, k = 2 \). Node 4, depicted here by partly filled circles, is a faulty node.

In addition, by (5.1)-(5.2), we can state that if \( \alpha_i \geq \alpha^*_i(n, k, d, \gamma, \xi, M) \) the point corresponding to parameters \( (n, k, d, \alpha_0, \delta, \beta, \xi, M) \) is feasible, and there exist linear codes to achieve that point. Theoretically, we cannot achieve points if \( \alpha_i < \alpha^*_i(n, k, d, \gamma, \xi, M) \). The function \( \alpha^*_i(n, k, d, \gamma, \xi, M) \) is computed as follows:

\[
\alpha^*_i = \begin{cases} 
\frac{M}{g(i)} & \gamma \in [h(0), \infty) \\
\frac{M - \gamma g(i)}{(k - i)(\xi - \delta) + \gamma(1 - \xi + \delta)} & \gamma \in [h(i), h(i-1)),
\end{cases}
\]

where

\[
g(i) = \frac{(2d - 2k + i + 1)i}{2d}, \quad \gamma \in [h(0), \infty)
\]

\[
h(i) = \frac{M}{g(i) + \left((k - i) + \frac{(1 - i + \delta)}{\xi - \delta}\right)(1 - \frac{k - 1 - \delta}{d})}, \quad \gamma \in [h(i), h(i-1)).
\]

The detailed proof of this theorem is provided in Section 5.3. Two extreme points on the tradeoff are derived as follows. For the minimum storage partial regenerating (MSPR) codes,

\[
\alpha_0 = \alpha_i = \frac{M}{k}, \\
\delta = 0, \\
\gamma_{\text{MSPR}} = \frac{M d \xi}{k(d - k + 1)}.
\]
5.2. Storage-Bandwidth Tradeoff in Partial Repair

For the minimum bandwidth partial regenerating (MBPR) codes,

\[
\alpha_o = \frac{M}{k} + \xi, \quad \alpha_i = \frac{M}{k}, \quad \delta = \xi, \quad \gamma_{MBPR} = 0.
\]  

(5.6)

One important observation on the bandwidth-storage tradeoff is that the codes on two extreme points can be constructed by MDS codes. At MSPR point, only one layer of MDS code is used. At MBPR point, first the file is encoded by an \((n, k)\)-MDS code and then in each storage node the packets are encoded by an \((\alpha_o, \alpha_i)\)-MDS code. Since there have been lots of research on efficient MDS erasure encoding and decoding in the literature (e.g., see [Rot06] and references therein), these codes can be efficiently used in this application. Another important point is that the partial-repair bandwidth approaches zero as the storage per node increases.

As an example, the feasible region and the optimal bandwidth-storage tradeoff for \(n = 15, k = 10, d = 14, \xi = 0.8\) and \(M = 10\) has been shown in Fig. 5.3. In this figure, Region I, whose border is characterized by the blue line with circles, is the feasible region where we only use outer-layer coding (which is similar to the tradeoff for regenerating codes in [DGW+10]). Region II, whose border is characterized by the red and bold line, is the feasible region where we use \((n, k)\)-MDS code as the

Figure 5.3: Bandwidth-storage tradeoff in partial-repair. On the new tradeoff, for the points on the curve between A and B, using only outer layer encoding is optimal, while for the points from B to C, a combination of inner layer and outer layer encoding is optimal.

For the minimum bandwidth partial regenerating (MBPR) codes,

\[
\alpha_o = \frac{M}{k} + \xi, \quad \alpha_i = \frac{M}{k}, \quad \delta = \xi, \quad \gamma_{MBPR} = 0.
\]  

(5.6)
outer layer code and then increase the inner-layer code rate. Region III, which is depicted in Fig. 5.3 by dashed lines, is the feasible region when a combination of inner and outer-layer codes is used for encoding the source file in the storage nodes. A convex combination (as defined later) of feasible points of Region I and II is a feasible point, as we formally state in the following proposition.

**Proposition 5.1.** Suppose \( \theta = \delta \alpha_i \) is redundant data in each storage node. If points \((n, k, d, \alpha_{i_1}, \theta_1, \gamma_1)\) and \((n, k, d, \alpha_{i_2}, \theta_2, \gamma_2)\) are feasible, then their convex combination, which is defined as the point \((n, k, d, \lambda \alpha_{i_1} + (1 - \lambda) \alpha_{i_2}, \lambda \theta_1 + (1 - \lambda) \theta_2, \lambda \gamma_1 + (1 - \lambda) \gamma_2)\) is also feasible, for \(0 < \lambda < 1\).

**Proof.** Since the points \((n, k, d, \alpha_{i_1}, \theta_1, \gamma_1)\) and \((n, k, d, \alpha_{i_2}, \theta_2, \gamma_2)\) are feasible then we have

\[
M \leq \sum_{i=0}^{k-1} \min \left[ (\xi \alpha_{i_1} - \theta_1), (1 - \frac{i}{d}) \gamma_1 \right] + k(\alpha_{i_1} - \xi \alpha_{i_1} + \theta_1),
\]

for \(l = 1, 2\). Thus, we have

\[
\lambda M \leq \sum_{i=0}^{k-1} \lambda \min \left[ (\xi \alpha_{i_1} - \theta_1), (1 - \frac{i}{d}) \gamma_1 \right] + k\lambda(\alpha_{i_1} - \xi \alpha_{i_1} + \theta_1),
\]

for \(0 < \lambda < 1\). Thus,

\[
\lambda M + (1 - \lambda) M \leq \sum_{i=0}^{k-1} \min \left[ (\lambda \xi \alpha_{i_1} - \lambda \theta_1), (1 - \frac{i}{d}) \lambda \gamma_1 \right] + k\lambda(\alpha_{i_1} - \xi \alpha_{i_1} + \theta_1) + \\
\min \left[ (1 - \lambda) \xi \alpha_{i_2} - (1 - \lambda) \theta_2, (1 - \frac{i}{d}) (1 - \lambda) \gamma_2 \right] + k(1 - \lambda)(\alpha_{i_2} - \xi \alpha_{i_2} + \theta_2).
\]  

(5.7)

Since for \(a, b, c, d\), four nonnegative real numbers, we have \(\min(a, b) + \min(c, d) \leq \min(a + c, b + d)\), then

\[
M \leq \sum_{i=0}^{k-1} \min \left[ (\lambda \alpha_{i_1} + (1 - \lambda) \alpha_{i_2}) - (\lambda \theta_1 + (1 - \lambda) \theta_2), \\
(1 - \frac{i}{d})(\lambda \gamma_1 + (1 - \lambda) \gamma_2) \right] + k((\lambda \alpha_{i_1} + (1 - \lambda) \alpha_{i_2}) - \xi(\lambda \alpha_{i_1} + (1 - \lambda) \alpha_{i_2}) + \\
(\lambda \theta_1 + (1 - \lambda) \theta_2)).
\]

Thus, point \((n, k, d, \lambda \alpha_{i_1} + (1 - \lambda) \alpha_{i_2}, \lambda \theta_1 + (1 - \lambda) \theta_2, \lambda \gamma_1 + (1 - \lambda) \gamma_2)\) is also feasible. \(\square\)
Finally, since the points on the bandwidth-storage tradeoff satisfy all the cut constraints in the information flow graph, we can design codes by linear random network codes [HMK+06], or deterministic network codes [JSC+05], where the codes have sufficiently large alphabet size.

5.3 Proof of the Main Result

5.3.1 Proof of Theorem 5.1

Our proof is inspired by the method adopted in [DGW+10]. We first show that an upper bound of the information rate in the information flow graph equals to

$$\sum_{i=0}^{k-1} \min \left[ \alpha_i, \left(1 - \frac{i}{d}\right)\gamma + \alpha_i(1 - \xi + \delta) \right].$$  \hspace{1cm} (5.9)

Then, we use the results in network coding literature [ACLY00, Yue08] to prove the achievability. The information flow graph after $k$ partial failure/repair processes can be shown in $k$ subsequent stages, as depicted in Fig. 5.4. We first prove that there is a cut in the network with the capacity given in (5.9). To this aim, we consider a cut that passes a route with a minimum capacity at any stage of repair. That is, at stage 1, the cut selects a route between $\alpha_i$ and $(d - 1)\beta + (1 - \xi + \delta)\alpha_i$. At stage 2, since the new node can obtain $\beta$ packets from the previously recovered node, then the cut selects between $\alpha_i$ and $(d - 1)\beta + (1 - \xi + \delta)\alpha_i$, and so on. Finally, there will be a graph with a cut capacity equivalent to (5.9). This is the min-cut in the information flow graph which can be achieved by linear network codes.

Next, we characterize the achievable tradeoffs between the storage $\alpha_o = \alpha_i(1 + \delta)$ and the partial repair bandwidth, $\gamma = d\beta$. To derive the optimal tradeoffs, we can fix the partial-repair bandwidth and also fix $\delta$ and then solve for the minimum $\alpha_i$ such that (5.9) is satisfied. We thus have the following optimization problem:

$$\alpha_i^* = \min_{\gamma} \alpha_i$$

s. t. \hspace{1cm} \sum_{i=0}^{k-1} \min \left[ \alpha_i, \left(1 - \frac{i}{d}\right)\gamma + \alpha_i(1 - \xi + \delta) \right] \geq M. \hspace{1cm} (5.10)

The optimization problem (5.10) can be explicitly solved. We call the solution $\alpha_i^*$, which for fixed $d$, and $\delta$ is a piecewise linear function over $\gamma$.

To simplify notation, we introduce

$$b_i \triangleq (1 - \frac{k - 1 - i}{d})\gamma_i, \text{ for } i = 0, \ldots, k - 1. \hspace{1cm} (5.11)$$

Then the problem is to minimize $\alpha_o$ subject to the constraint

$$\sum_{i=0}^{k-1} \min\{b_i + \alpha_i(1 - \xi + \delta), \alpha_i\} = M, \hspace{1cm} (5.12)$$
which is equivalent to the following constraint that
\[
\sum_{i=0}^{k-1} \min(b_i, \alpha_i(\xi - \delta)) + k\alpha_i(1 - \xi + \delta) = M. \tag{5.13}
\]

The left hand side of (5.12), denoted as \( C(\alpha_i) \), is piecewise linear function over \( \alpha_i \).

\[
C(\alpha_i) = \begin{cases} 
  k\alpha_i & (\xi - \delta)\alpha_i \in [0, b_0], \\
  b_0 + ((\xi - \delta)(k - 1) + k(1 - \xi + \delta))\alpha_i & (\xi - \delta)\alpha_i \in (b_0, b_1], \\
  \vdots & \\
  b_0 + b_1 + \cdots + b_{k-1} + k(1 - \xi + \delta)\alpha_i & (\xi - \delta)\alpha_i \in (b_{k-1}, \infty].
\end{cases} \tag{5.14}
\]

Then we find the minimum \( \alpha_i \) as \( \alpha_i^* = C^{-1}(M) \). That is,
\[
\alpha_i^* = \begin{cases} 
  \frac{M}{k} & M \in \left[0, \frac{k b_0}{(1 - \xi + \delta)}\right], \\
  \frac{M - b_0}{k - 1 + k(1 - \xi + \delta)} & M \in \left(b_0 + \frac{(k - 1)(1 - \xi + \delta)}{(1 - \xi + \delta)} b_0, b_0 + \frac{(k - 1)(1 - \xi + \delta)}{(1 - \xi + \delta)} b_1\right], \\
  \vdots & \\
  \frac{M - \sum_{j=0}^{k-1} b_j}{k(1 - \xi + \delta)} & M \in \left(\sum_{j=0}^{k-1} b_j + \frac{(k - 1)(1 - \xi + \delta)}{(1 - \xi + \delta)} b_{k-1}, \infty\right].
\end{cases} \tag{5.15}
\]

In general we have,
\[
\alpha_i^* = \frac{M - \sum_{j=0}^{i-1} b_j}{(k - i)(\xi - \delta) + k(1 - \xi + \delta)}, \tag{5.16}
\]
5.4. Conclusions

when

\[ M \in \left( \sum_{j=0}^{i-1} b_j + \left( \frac{k(1 - \xi + \delta)}{(\xi - \delta)} \right) b_{i-1}, \sum_{j=0}^{i-1} b_j + \left( \frac{k(1 - \xi + \delta)}{(\xi - \delta)} \right) b_i \right). \]

Considering that

\[ \sum_{j=0}^{i-1} b_j + \left( \frac{k(1 - \xi + \delta)}{(\xi - \delta)} \right) b_{i-1} = \sum_{j=0}^{i-2} b_j + \left( \frac{k(1 - \xi + \delta)}{(\xi - \delta)} \right) b_{i-1}, \]

we define

\[ h(i) \equiv M \gamma \left( \sum_{j=0}^{i-1} b_j + \left( \frac{k(1 - \xi + \delta)}{(\xi - \delta)} \right) b_i \right)^{-1}. \]

We then have

\[ M \in \left( \frac{M \gamma}{h(i-1)}, \frac{M \gamma}{h(i)} \right). \]

Since \( \sum_{j=0}^{i-1} b_j = \gamma g(i) \) then

\[ \alpha^*_i = \left( \frac{M}{\gamma (k-i)(\xi-\delta)+k(1-\xi+\delta)} \right), \quad \gamma \in [0, \infty), \gamma \in [h(i), h(i-1)]. \]

5.4 Conclusions

We studied the partial repair problem in distributed storage systems, in which the stored packets in the storage nodes may be lost. We proposed two layer coding for storing the file. We showed that the partial repair bandwidth can be reduced by two-layer coding. We derived the optimal storage-bandwidth tradeoff for two layer coding. The tradeoff can be achieved by linear network codes.
Security in Distributed Storage Systems with Partial Node Failure/Repair

In this chapter, we consider a distributed storage system where parts of the stored file fragments in storage nodes may be lost. In the previous chapter, there was only one storage node that lost a part of its stored data. In this chapter, there can be multiple nodes that lost their stored data. We denote a storage node that lost a part of its fragments as a faulty storage node and a storage node that did not lose any fragment as a correct storage node. In a process, termed as partial repair, a set of storage nodes (among faulty and correct storage nodes) transmit repairing fragments to the faulty storage nodes. We first investigate the optimal partial repair in which the required bandwidth for recovering the lost fragments is minimum. Next, we assume that an eavesdropper wiretaps a subset of links between storage nodes, and overhears a number of repairing fragments. We then study secure partial-repair in which the eavesdropper obtains no information from the repairing fragments. We propose codes that are optimal in repair-bandwidth and are also optimal in terms of strong or weak security conditions. We also provide optimal secure codes for exact partial-repair in a scenario. We show the gain of our proposed codes compared to random network codes in achieving the optimal security bounds.

6.1 Introduction

In distributed storage systems which include centralized/distributed cloud storage systems, peer-to-peer cloud storage systems and private/public data centers, users can store, archive, or back up their data on the (geographically) distributed storage nodes. DropBox [DMMM^12], Google File Systems [GGL03], and AmazonS3 [PIRG08] are among the examples of these storage systems. There are two main concerns in these systems. One concern is on security. That is, in geographically distributed systems where the Internet is used for communication between two physically separated nodes, the transmitted information between a pair of nodes might be overheard by an eavesdropper [CBR03 SL00]. Another
Figure 6.1: A distributed storage system, where the system stores a file containing four fragments $a_1, a_2, b_1, b_2$ by a $(4, 2)$-MDS code over the finite field $\mathbb{F}_3$. Suppose that, fragments $a_1 + a_2$ in node 3 and $b_2$ in node 4 are lost. To recover the lost fragments, we can consider the faulty nodes as completely failed nodes. Then by regenerating codes in $\text{DGW}^{10}$, we must transmit the entire file (four fragments) to each faulty node. This requires eight fragment transmissions in total. A better approach, proposed in $\text{KLSS}^{11}, \text{HXW}^{10}, \text{SH}^{13}$, is to allow the completely failed nodes to cooperate. In this case, six fragment transmissions in total are required (for details, please see $\text{KLSS}^{11}, \text{HXW}^{10}, \text{SH}^{13}$). Alternatively, one can consider each fragment in the system as a (virtual) node; then each lost fragment requires two fragment transmissions (e.g., for recovering $b_2$ in node 4, node 1 sends $b_1$ and node 2 sends $2b_1 + b_2$ to node 4). Thus, in total four fragment transmissions are required to recover all the lost fragments. In our proposed approach, we exploit the available side information in the faulty nodes. Then, node 1 and 2 respectively send $a_1 + b_1$ and $2(a_1 + b_1) + a_2 + b_2$ toward node 4. Next, node 4 can recover its lost fragments by the received fragments and its side information, after performing some operations over $\mathbb{F}_3$ as $2(a_1 + b_1) + a_2 + b_2 - 2(a_1 + b_1) - a_2 = b_2$. Node 4 then sends coded fragment $a_1 + b_1 + a_2 + b_2$ toward node 3. Then, node 3 recovers its lost fragment by an operation in $\mathbb{F}_3$ as $a_1 + b_1 + a_2 + b_2 - (b_1 + b_2) = a_1 + a_2$. Therefore, the lost fragments are recovered by three fragment transmissions. We show later that three number of transmissions is the minimum required total number of transmissions for partial repair in this example.
6.1. Introduction

When an eavesdropper overhears the links in the Internet, it may obtain some information about the stored files in distributed storage systems. To provide strong security, we store $z_1$ and $z_2$, two random symbols taken uniformly from $\mathbb{F}_3$ and then send coded fragments using those random symbols. By that, the eavesdropper cannot obtain any information about the source file while overhearing the repairing fragments. Here, security is achieved with a cost of losing some storage space for storing those random symbols.

Maximum distance separable (MDS) codes provide the highest reliability for the given storage space. A file when coded by an $(n, k)$-MDS code is divided into $k$ equal-sized blocks which are then coded to $n$ blocks such that any set of $k$ blocks can reconstruct the original source file. Most of the existing studies provide codes considering node failure, where all data in a storage node is lost [DGW+10, WD09, RSKR09b, GXS11, KLSS11]. Then, in a process, termed as the repair process, a new node is generated by the help of surviving nodes. For the repair, the optimal bandwidth is studied in [DGW+10], where the use of network coding is proposed in repair. After repair, the new node might contain different data compared to the failed node. But, it preserves the property that every $k$ storage nodes can reconstruct the source file. This repair is termed as the functional repair. In contrast, in exact repair the content of the new node is the same as the failed node. Exact repair has been studied in [RSKR09b]. To further decrease the repair bandwidth, cooperative repair has been proposed in [KLSS11, HXW+10, SH13] when multiple nodes fail. Applying the existing repair methods (which assume complete node failure) to the case of partial loss might be suboptimal, as it is illustrated in Fig. 6.1. In a recent work in [GXS15], the partial repair problem has been studied in wireless caching networks with broadcast channels. We study partial repair problem in distributed storage systems, where channels between storage nodes are point-to-point channels. In another work in [BHH13], partial-MDS codes have been studied for the systems that parts of data in storage node are lost. However, in [BHH13] bandwidth in partial-repair has not been considered. We focus on designing partial-MDS codes that require the minimum partial-repair bandwidth. Our proposed codes efficiently exploit the available side

---

\footnote{Here, a block may contain a number of equal-sized fragments of information.}
information in faulty nodes to achieve the optimal partial-repair. We also propose codes for the exact partial-repair in a scenario, in which faulty nodes equally lose a fraction of their stored fragments.

With the optimal bandwidth in partial repair, we study security in partial-repair by an information-theoretic perspective. Clearly, it is important to avoid any leakage of information to an unintended user, especially where storage nodes are geographically distributed and the Internet is used for communication between nodes. We propose secure codes for partial repair such that an eavesdropper obtains no information by overhearing a finite number of repairing fragments. In particular, we propose codes that are simultaneously optimal in security and in partial-repair bandwidth. For that, we define the secrecy capacity as the maximum amount of information that can be stored in the system such that the eavesdropper cannot obtain any information about the source file. Our work is related to secure network codes in multicast networks. The secrecy capacity of multicast networks was derived in [CY02], where an eavesdropper overhears a subset of links. In [PERR11, HSHC13, HPZM13], security in the repair problem of distributed storage systems was studied. The previous studies of the security in the repair problem mostly assumed (complete) node failure. That is, all the stored fragments in a failed node are lost [PERR11]. In a recent work, the authors in [GXSS15] studied security in partial repair in wireless caching networks with broadcast channels. The study is different from [GXSS15] in the senses that i) we study the optimal secure codes in distributed storage system where storage nodes are connected by point-to-point channels and there is no broadcast channels. Thus, there is a cost associated with transmitting repairing fragments between a pair of storage nodes. ii) we study security for bandwidth-optimal codes where the eavesdropper overhears a subset of links instead of overhearing all the repairing fragments (in contrast to [GXSS15]). To achieve the secrecy capacity, the source file fragments are encoded (i.e., precoded) before using MDS codes with the optimal partial repair-bandwidth. We derive the sufficient field size for codes achieving the secrecy capacity. Further, we provide deterministic codes for secure and exact partial repair. We show how our deterministic codes outperform random codes over a large finite field size.

Our contribution can be summarized as follows. We study the optimal partial repair in distributed storage systems, and then investigate secure codes for the optimal partial-repair. Our optimal partial-repair codes can be a generalized form of cooperative repair [KLSS11, HXW+10, SH13], as we describe it more later. We also find secrecy capacity in the senses of information-theoretically strong and weak security notions over this generalized form of repair. Since exact repair is also interesting in practice, we propose exact partial-repair codes. The exact partial-repair codes and the security of the exact partial-repair are studied for the first time.

The rest of the chapter is organized as follows. In Section 6.2, we formulate the optimal secure partial-repair problem. In Section 6.3, we provide our main results. In Section 6.4, we present explicit code construction for exact and secure partial-repair. We discuss about the impacts of loss patterns and random codes in
Section 6.2

Finally, we conclude the chapter in Section 6.6.

6.2 Problem Formulation

Before we formally introduce the problem, we describe the notation.

**Notation:** We use a bold lowercase letter to denote a column vector, and a bold uppercase letter to denote a matrix. Superscript $T$ denotes matrix transpose. The set $[n]$ denotes the set of integers from 1 to $n$, i.e., $[n] = \{1, 2, \ldots, n\}$. $|\mathcal{P}|$ denotes the cardinality of the set $\mathcal{P}$. For a random variable $X$, $H(X)$ denotes the entropy of $X$.

We consider a distributed storage system with $n$ storage nodes. The source file of size $M$ is encoded by an $(n, k) - \text{MDS code}$. That is, any set of $k$ storage nodes can reconstruct the source file. We may refer to this property as the reconstruction property. The source file contains $M$ fragments which are elements of $\mathbb{F}_q$, and $q$ denotes the code alphabet size. If we denote the source file by a column vector $s = (s_1, s_2, \ldots, s_M)$, then a coded fragment $x$ is formulated by $x = g^T s$. The vector $g$, known as global encoding vector [Ye08]. We say two fragments $x_1 = g_1^T s$ and $x_2 = g_2^T s$ are independent if their global encoding vectors $g_1$ and $g_2$ are independent.

When parts of the stored fragments in one or more storage nodes are lost and the faulty storage nodes have no access to the source file, storage nodes exchange information to recover the lost fragments. The process of recovering the lost fragments is called partial repair. To formulate the partial repair process, assume that a storage node $i$ has access to a set $\mathcal{P}_i$ of $|\mathcal{P}_i|$ independent fragments, i.e., $p_{i1}, p_{i2}, \ldots, p_{i|\mathcal{P}_i|}$. Subsequently, the superset $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n\}$ indicates the number of available fragments on each storage node. Clearly, we can also deduce the number of lost fragments in node $i$ is $M/k - |\mathcal{P}_i|$.

Clearly, to recover all the lost fragments in a partial-repair process, it is necessary that the available information in the system is not less than $M$. Let $P = (p_{11}, p_{12}, \ldots, p_{1|\mathcal{P}_1|}, \ldots, p_{n1}, p_{n2}, \ldots, p_{n|\mathcal{P}_n|})$ denote the random variable associated with the available fragments in $n$ storage nodes, then we formally state the necessary condition over a given loss pattern $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n\}$ as

$$H(P) \geq M.$$  \hfill (6.1)

Otherwise some information is permanently lost and thus the repair is not possible. We always assume (6.1) holds.

The transmission schedule in partial repair can be explicitly stated in the following two steps:

1. Correct storage nodes transmit fragments (functions of their stored fragments) to the faulty storage nodes;
2. Faulty storage nodes transmit fragments (functions of their stored fragments and their received fragments) to other faulty storage nodes.

Note that partial repair is different from cooperative repair studied in \cite{KLSS11, HXW+10, SH13} in the two following senses. In our partial repairing process:

1. Storage nodes are allowed to transmit different amount of data on different links, and

2. There is side information available on faulty storage nodes.

With an abuse of notation, we say in partial repair, storage node \(i\) transmits \(\beta_{ij}\) fragments to storage node \(j\) for \(i, j \in [n]\), considering that \(\beta_{ij} = 0\) for \(i = j\). In addition, following to the above notation and our strategy in partial repair, \(\beta_{ij} = 0\), if node \(j\) is a correct storage node (there is no transmission to a correct storage node). We denote the total number of transmissions in a partial-repair process by \(\Gamma\), and is termed as partial-repair bandwidth. Formally,

\[
\Gamma \triangleq \sum_{i,j=1}^{n} \beta_{ij}. \quad (6.2)
\]

We define \(R(\mathcal{P})\), the feasible region for partial repair, as follows.

**Definition 6.1.** Consider a distributed storage system that stores a file of size \(M\) by an \((n, k)\)-MDS code. Suppose some fragments are lost and the available data on storage nodes is based on the set \(\mathcal{P} = \{P_1, P_2, \ldots, P_n\}\). Define the region \(R(\mathcal{P}) \subset \mathbb{R}^{n \times n}\) to be set of all \(\{\beta_{ij}\}\) satisfying

\[
\sum_{j \in Q} \sum_{i \in Q^c} \beta_{ij} \geq M - \sum_{j \in Q} |P_j|, \quad (6.3)
\]

for every \(Q \subset [n]\) and \(|Q| = k\) storage nodes. In the next section, we show that \(R(\mathcal{P})\) characterizes the necessary and sufficient conditions over \(\beta_{ij}\)'s for successful partial-repair.

Next, we formulate security in partial repair as follows. Let \(Y_1, Y_2, \ldots, Y_\Gamma\) denote the random variables representing \(\Gamma\) transmitted repairing fragments. Now assume that there is an eavesdropper who wiretaps a subset \(\mathcal{E}\) of the repairing fragments and has access to \(\mu\) independent fragments (fragments with independent global encoding vectors) and \(\mu \leq |\mathcal{E}|\). We aim to design partial repair codes such that there would be no leakage of information to the eavesdropper, in terms of strong and weak security conditions, which are defined as follows.

**Definition 6.2 (Strong Security).** Consider a distributed storage system in which the source file is distributed among \(n\) storage nodes. Let \(S\) denote the random variable associated with the source file fragments \((s_1, s_2, \ldots, s_M)\), and \(E\) denote...
the random variable associated with \(|E|\) fragments \((e_1, e_2, \ldots, e_{|E|})\), observed by the eavesdropper. The partial repair is strongly secure, if

\[ H(S|E) = H(S) \] (6.4)

**Definition 6.3 (Weak Security).** Consider a distributed storage system in which a source file is distributed among \(n\) storage nodes. Let \(S_i\) denote the random variable associated with \(i\)-th fragment of the source file. The partial repair is weakly secure, if

\[ H(S_i|E) = H(S_i) \] for \(i = 1, \ldots, M.\) (6.5)

Unlike the strong security condition, the eavesdropper in the weak security condition obtains some information, but it cannot deduce any meaningful information about the individual fragments.

By the above security definitions, we aim to find the maximum amount of information that can be stored in the storage system such that the strong or weak secrecy conditions are satisfied. More formally, suppose that we use an \((n, k)\)-MDS code for storing the source file in the storage nodes, then a set of \(k\) nodes can reconstruct the source file. That is, if \(D\) denotes the random variable associated with the fragments in a set \(D\) of \(k\) storage nodes, then \(H(S|D) = 0\). In addition, the eavesdropper overhearing fragments \(Y_1, \ldots, Y_{|E|}\) obtains no information about the source if

\[ H(S|Y_1, \ldots, Y_{|E|}) = H(S). \] (6.6)

We formally define the strong secrecy capacity (which is here denoted as \(C_{ss}\)) as

\[ C_{ss} \triangleq \max \quad H(S), \]
\[ \text{subject to:} \quad H(S|Y_1, \ldots, Y_{|E|}) = H(S), \]
\[ H(S|D) = 0, \quad \text{for } \forall D \subset [n], |D| = k. \] (6.7)

Following the same setting when the eavesdropper overhears fragments \(Y_1, \ldots, Y_{|E|}\), we formally define the weak secrecy capacity. We denote the weak secrecy capacity as \(C_{ws}\), then we have

\[ C_{ws} \triangleq \max \quad H(S), \]
\[ \text{subject to:} \quad H(s_i|Y_1, \ldots, Y_{|E|}) = H(s_i), \]
\[ \text{for } i \in \{1, \ldots, M\}, \]
\[ H(S|D) = 0, \quad \text{for } \forall D \subset [n], |D| = k. \] (6.8)

We shall study the strong and weak secrecy capacities in the next sections.

### 6.3 Main Results

In this section, we provide the main results of this chapter. We first formulate an optimization problem to derive the minimum required bandwidth in partial-repair.
Next, we derive strong and weak secrecy capacities. We note that the results in this section is valid for functional partial-repair.

Our first contribution characterizes the feasible region for partial repair. The following theorem states the necessary and sufficient conditions for partial repair.

Theorem 6.1. Consider a distributed storage system using an \((n, k)\)-MDS code. Suppose the storage nodes have access to parts of their stored data based on \(\mathcal{P} = \{P_1, P_2, \ldots, P_n\}\). For given \(\beta_{ij}\)'s, then all the lost fragments can be functionally recovered if and only if \(\{\beta_{ij}\}_{i,j=1}^n \in R(\mathcal{P})\).

Proof. The detailed proof is provided in Appendix A.

We will apply this theorem on our running example in Fig. 6.3.

Example 6.1. Examining Theorem 6.1 on the four-node storage network in Section 6.2 provides us the following conditions for partial repair:

In general, the optimal bandwidth for exact partial-repair is unknown.
6.3. Main Results

Table 6.1: The feasible region in Example 1

<table>
<thead>
<tr>
<th>Q</th>
<th>induced constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1, 3}</td>
<td>(\beta_{23} + \beta_{43} \geq 1)</td>
</tr>
<tr>
<td>{1, 4}</td>
<td>(\beta_{24} + \beta_{34} \geq 1)</td>
</tr>
<tr>
<td>{3, 4}</td>
<td>(\beta_{13} + \beta_{23} + \beta_{14} + \beta_{24} \geq 2)</td>
</tr>
<tr>
<td>{2, 3}</td>
<td>(\beta_{43} + \beta_{13} \geq 1)</td>
</tr>
<tr>
<td>{2, 4}</td>
<td>(\beta_{34} + \beta_{14} \geq 1)</td>
</tr>
</tbody>
</table>

Note that for \(Q\) as a set of \(k\) correct storage nodes, e.g., here for \(Q = \{1, 2\}\), we have \(M - \sum_{j \in Q} |P_j| = 0\). Then, for these cases, there would not be any constraint on \(\beta_{ij}\)'s. We may call these as inactive constraints. For analysis, we can only consider active constraints. Note also that some \(\beta_{ij}\)'s are always zero. For example, here, \(\beta_{ij} = 0\) if node \(i\) and \(j\) are two correct storage nodes. Finally, summing both sides of the active constraints over non-zero \(\beta_{ij}\)'s gives us \(2(\beta_{43} + \beta_{13} + \beta_{23} + \beta_{34} + \beta_{14} + \beta_{24}) \geq 6\). This turns out that \(\Gamma = \beta_{43} + \beta_{13} + \beta_{23} + \beta_{34} + \beta_{14} + \beta_{24} \geq 3\), and thus the minimum required bandwidth for partial repair is 3. Thus, the repairing code presented in Fig. 6.1 is optimal. Following quite the same arguments, we can derive the lower bound of the required bandwidth in partial-repair for more general cases. This is stated in the following proposition.

Proposition 6.1. Consider a distributed storage system where a file of size \(M\) is encoded by an \((n, k)\)-MDS code. Suppose that \(n - k\) storage nodes have equally lost \(\xi\) fragments of their stored fragments, where \(0 \leq \xi \leq M/k\). We assume that there are always \(k\) correct storage nodes in the system. This assumption assures us that the file availability condition in (6.7) is always satisfied, for any value of \(\xi\), e.g., even for \(\xi = M/k\). Without loss of generality, we assume that nodes \(1, 2, \ldots, k\) are correct storage nodes and the other \((n - k)\) storage nodes are faulty storage nodes. That is, \(|P_1| = M/k\), for \(i \in \{1, \ldots, k\}\), and \(|P_i| = M/k - \xi\), for \(i \in \{k + 1, \ldots, n\}\). A lower bound on the partial repair-bandwidth is

\[
\Gamma \geq (n - 1)\xi.
\]  

(6.10)

Proof. The proof is provided by generalizing the approach given in Example 1. Before providing the technical proof, let us state a useful lemma.

Lemma 6.1. For \(n\), and \(k\) non-zero integer values and \(n > k\), we have

\[
\sum_{i=1}^{k} \binom{k}{i} \binom{n-k}{i} = \frac{k(n-k)!}{(k-1)!(n-k-1)!}
\]  

(6.11)
By Theorem 1, to have a successful repair for any set \( D \) of \( k \) nodes, the incoming fragments to the set have to be greater than \( M - \sum_{i \in D} |P_i| \). We thus examine the cut-set bound analysis for any selection of \( k \) out of \( n \) nodes, as follows. If all \( k \) nodes are perfect nodes, then there is no need for repairing fragments. For other number of faulty nodes among \( k \) nodes we have these conditions:

- If one faulty node is in \( D \), then
  \[
  \sum_{i \in [n] \setminus D} \sum_{j \in D} \beta_{ij} \geq \xi \tag{6.12}
  \]
  We have \( \binom{k}{1} n \binom{n-k}{1} \) such these cases.

- If two faulty nodes are in \( D \), then
  \[
  \sum_{i \in [n] \setminus D} \sum_{j \in D} \beta_{ij} \geq 2\xi \tag{6.13}
  \]
  We have \( \binom{k}{2} n \binom{n-k}{2} \) such these cases.

- We continue the same approach until there are \( k \) faulty nodes in \( D \).

- If \( k \) faulty nodes are in \( D \), then
  \[
  \sum_{i \in [n] \setminus D} \sum_{j \in D} \beta_{ij} \geq k\xi \tag{6.14}
  \]
  We have \( \binom{k}{k} n \binom{n-k}{k} \) such these cases.

We now sum up both sides of the above inequalities. Since each term \( \beta_{ij} \) appears \( \binom{n-2}{k-1} \) times, then we have

\[
\binom{n-2}{k-1} \Gamma \geq \xi\binom{k}{1} \binom{n-k}{1} + 2\xi \binom{k}{2} \binom{n-k}{2} + \cdots + k\xi \binom{k}{k} \binom{n-k}{k}
= \xi \sum_{i=1}^{k} \binom{k}{i} \binom{n-k}{i} \tag{6.15}
= \xi \binom{(n-1)!}{(k-1)! \cdot (n-k-1)!}, \tag{6.16}
\]
where (6.16) holds by Lemma 6.1. Hence, we have

\[
\Gamma \geq (n - 1)\xi. \tag{6.17}
\]

This finalizes the proof.
6.3. Main Results

In general, we aim to minimize partial repair-bandwidth, \( \Gamma = \sum_{i,j=1}^{n} \beta_{ij} \), for the points in the feasible region. Formally, we aim to find

\[
\min_{\beta_{ij}} \quad \Gamma = \sum_{i,j=1}^{n} \beta_{ij}
\]

subject to:

\[
\sum_{j \in Q} \sum_{i \in Q_{j}} \beta_{ij} \geq M - \sum_{i \in Q} |P_{i}|, \quad Q \subset [n], |Q| = k, \quad \beta_{ij} \in \mathbb{R}^{+},
\]

where \( \mathbb{R}^{+} \) is the set of non-negative real numbers. This problem is a linear programming problem and can be efficiently solved [BV09, PS98].

6.3.1 Strong Secrecy Capacity

For a given loss pattern \( \mathcal{P} = \{P_1, P_2, \ldots, P_n\} \), we can derive the strong secrecy capacity, as follows.

**Theorem 6.2.** Consider a distributed storage system where a file is coded by an \((n, k)\)-MDS code, and every set of \( k \) nodes can maximally store \( M \) fragments. Suppose that an eavesdropper overhears a subset \( \mathcal{E} \) of repairing fragments where the matrix of the global encoding vectors of \( |\mathcal{E}| \) fragments has rank \( \mu \leq |\mathcal{E}| \). Then the strongly secure storage capacity is

\[
C_{ss} = M - \mu.
\]

**Proof.**

\[
H(S) = H(S | Y_1, Y_2, \ldots, Y_{|\mathcal{E}|}) - H(S | D),
\]

\[
= I(S; D) - I(S; Y_1, Y_2, \ldots, Y_{|\mathcal{E}|}),
\]

\[
= H(D) - H(Y_1, Y_2, \ldots, Y_{|\mathcal{E}|})
\]

\[
- H(D|S) + H(Y_1, Y_2, \ldots, Y_{|\mathcal{E}|} | S)
\]

\[
\leq M - \mu.
\]

In the above equations, (6.20) holds because of the strong security condition \( H(S | Y_1, Y_2, \ldots, Y_{|\mathcal{E}|}) = H(S) \), and the fact that every set of \( k \) nodes can reconstruct the stored file, i.e., \( H(S | D) = 0 \). Since the eavesdropper has access to \( \mu \)
independent repairing fragments, we have \( H(Y_1, Y_2, \ldots, Y_{|E|}) = \mu \). In (6.22), we have \( H(Y_1, Y_2, \ldots, Y_{|E|}|S) - H(D|S) \leq 0 \) since

\[
H(D, Y_1, Y_2, \ldots, Y_{|E|}|S) = H(D|S) + H(Y_1, Y_2, \ldots, Y_{|E|}|D, S). \tag{6.25}
\]

Since \((Y_1, Y_2, \ldots, Y_{|E|})\) is a function of \(D\) and \(S\) then \(H(Y_1, Y_2, \ldots, Y_{|E|}|D, S) = 0\), and since \(H(D|S) \geq 0\), then \(H(Y_1, Y_2, \ldots, Y_{|E|}|S) - H(D|S) \leq 0\).

The upper bounds can be achieved for functional partial-repair. To prove, we use the following lemma.

**Lemma 6.2** (Theorem 2 in [ERS07]). Consider a multicast network with \(t\) destinations, the min-cut capacity \(R\), and the corresponding directed acyclic graph \(G(N, A)\) with unit capacity edges. Here, \(N\) is the set of nodes and \(A\) is the set of edges of the graph. If an eavesdropper overhears \(\mu\) edges, then the rate \(r < R - \mu\) can be achieved with strong security conditions by precoding the source data and using a coset code having alphabet size \(q\), where

\[
q \geq \left( \frac{|A| - 1}{\mu - 1} \right) + t. \tag{6.27}
\]

**Proof.** [Sketch] If \(H\) denotes the parity check matrix of a coset code, and \(C_W\) denotes the matrix of global encoding vectors of the packets observed by the eavesdropper, then the multicast data is strongly secure if

\[
\text{rank} \begin{pmatrix} H \\ C_w \end{pmatrix} = M \text{ for all } C_w, \text{ where } \text{rank } C_w = \mu. \tag{6.28}
\]

Then an iterative algorithm is used to find a secure network code on the graph by the following steps. Firstly, \(t\) flows from the source node to \(t\) destination nodes are found and the corresponding global encoding matrices at the destination nodes, denoted as \(B_1, B_2, \ldots, B_t\), are set to identity matrix, i.e., \(B_1 = I, \ldots, B_t = I\). Since the graph is acyclic, there is a topological order of nodes. Then, starting from the outgoing edges of the source node, at each edge the local coding coefficient is found such that global encoding matrices \(B_1, \ldots, B_t\) remain full rank. From [JSC+05] when there is no security constraint, this algorithm is successful if \(q > t\). To satisfy the strong security condition, at each iteration the matrix \(\begin{pmatrix} H \\ C_w \end{pmatrix}\) must also be full rank. Since at each iteration on one edge it remains \(\mu - 1\) other edges among \(|A| - 1\) that the eavesdropper can overhear, the algorithm must find local coding coefficients such that \(\left( \frac{|A| - 1}{\mu - 1} \right) + t\) number of matrices become full rank. Thus,

\[
q \geq \left( \frac{|A| - 1}{\mu - 1} \right) + t. \tag{6.29}
\]
is sufficient to have a strongly secure network code.

Now, we use the above lemma to prove the achievability in the partial repair problem. As shown in Theorem 6.1, the functional partial-repair can be modeled into a multicast network. Hence, we use coset codes before MDS source encoding, as shown in Fig. 6.4. Using the above lemma, there exist linear codes for achieving the strong secrecy capacity as long as

\[ q \geq \left( M - 1 \right) + \left( \frac{n}{k} \right), \]

(6.30)

using the fact that the maximum independent edges is \( M \), and at most there are \( \binom{n}{k} \) different destination nodes. This finalizes the proof.

In a specific case, when an eavesdropper overhears all the repairing fragments in an instance of partial repair, the upper bound of strong secrecy capacity can be stated in the following corollary.

**Corollary 6.1.** Suppose an eavesdropper has access to \( Y_1, Y_2, \ldots, Y_\Gamma \), i.e., all the the repairing fragments in one stage of partial repair, then

\[ C_{ss} \leq \min_{\mathcal{D} \subset [n], |\mathcal{D}| = k} \sum_{i \in \mathcal{D}} |P_i|. \]

(6.31)

**Proof.** We first prove that,

\[ H(Y_1, Y_2, \ldots, Y_\Gamma) \geq \max_{\mathcal{D} \subset [n], |\mathcal{D}| = k} M - \sum_{i \in \mathcal{D}} |P_i|. \]

(6.32)

According to Theorem 6.1 for partial repair, for any set \( \mathcal{D} \) containing \( k \) nodes, the number of independent fragments transmitted toward the set \( \mathcal{D} \) must not be less than \( M - \sum_{i \in \mathcal{D}} |P_i| \). In other words, suppose that \( Y_{i_1}, Y_{i_2}, \ldots, Y_{i_j} \) (where \( \{i_1, \ldots, i_j\} \subseteq \{1, \ldots, \Gamma\} \)) are transmitted fragments toward \( \mathcal{D} \), we have

\[ H(Y_1, Y_2, \ldots, Y_\Gamma) \geq H(Y_{i_1}, Y_{i_2}, \ldots, Y_{i_j}) \]

\[ \geq M - \sum_{i \in \mathcal{D}} |P_i|. \]

(6.33)

(6.34)

The inequality in (6.33) holds because the set \( \{Y_{i_1}, Y_{i_2}, \ldots, Y_{i_j}\} \) is a subset of \( \{Y_1, Y_2, \ldots, Y_\Gamma\} \). Since \( H(Y_1, Y_2, \ldots, Y_\Gamma) \geq M - \sum_{i \in \mathcal{D}} |P_i| \) for any \( \mathcal{D} \), then we have

\[ \mu = H(Y_1, Y_2, \ldots, Y_\Gamma) \geq \max_{\mathcal{D} \subset [n], |\mathcal{D}| = k} M - \sum_{i \in \mathcal{D}} |P_i|. \]

By substituting the lower bound for \( \mu \) in (6.23), we derive the upper bound of the strong secrecy capacity.
6.3.2 Weak Secrecy Capacity

The following theorem states the weak secrecy capacity.

**Theorem 6.3.** Consider a distributed storage system where a file is coded by an \((n, k)\)-MDS code, and every set of \(k\) node can maximally store \(M\) fragments. Suppose that an eavesdropper overhears a subset \(\mathcal{E}\) of repairing fragments where the matrix of the global encoding vectors of \(|\mathcal{E}|\) fragments has rank \(\mu \leq |\mathcal{E}|\). If \(\mu < M\) then weak secrecy capacity is \(M\); otherwise the weak secrecy capacity is zero.

**Proof.** As the system can store a file of size \(M\) by an MDS code, then \(C_{ws}\) has a trivial upper bound as,

\[
C_{ws} \leq M. \quad (6.35)
\]

The upper bound can be achieved for \(\mu < M\). For achieving the bound, we use the following lemma.

**Lemma 6.3.** Consider a multicast network with \(t\) destinations, the min-cut capacity \(R\), and the corresponding directed acyclic graph \(\mathcal{G}(\mathcal{N}, \mathcal{A})\) with unit capacity edges. Here, \(\mathcal{N}\) is the set of nodes and \(\mathcal{A}\) is the set of edges of the graph. If an eavesdropper overhears \(\mu\) edges, then the rate \(r < R\) with weak security conditions can be achieved by precoding the source data and using a coset code having alphabet size \(q\), where

\[
q \geq \left(\frac{|\mathcal{A}| - 1}{\mu - 1}\right)R + t. \quad (6.36)
\]

**Proof.** If \(H\) denotes the parity check matrix of a coset code, \(h_i\) denotes the \(i\)-th row of matrix \(H\) and \(C_w\) denotes the matrix of global encoding vectors of the fragments observed by the eavesdropper, then the partial repair is weakly secure if

\[
\text{rank} \begin{pmatrix} h_i \\ C_w \end{pmatrix} = \mu + 1 \text{ for } i = 1, \ldots, R, \text{ and for all } C_w, \quad (6.37)
\]

where \(\text{rank} \ C_w = \mu\). Then an iterative algorithm is used to find a secure network code on the graph by the following steps. Firstly, \(t\) flows from the source node to \(t\) destination nodes are found and the corresponding global encoding matrices at the destination nodes, denoted as \(B_1, B_2, \ldots, B_t\), are set to identity matrix, i.e., \(B_1 = I, \ldots, B_t = I\). To satisfy the weak security conditions, at each iteration the matrix \( \begin{pmatrix} h_i \\ C_w \end{pmatrix} \) must also be full rank. Since at each iteration on one edge it remains \(\mu - 1\) other edges among \(|\mathcal{A}| - 1\) that the eavesdropper can overhear, and also there are at most \(R\) number of different \(h_i\)'s, the algorithm must find local
coding coefficients such that \( (|A|-1)R + t \) number of matrices become full rank. Thus,

\[
q \geq \left( \frac{|A| - 1}{\mu - 1} \right) R + t \tag{6.38}
\]

is sufficient to have a weakly secure network code.

Now, we use the above lemma to prove the achievable in the partial repair problem. We use coset codes before MDS source encoding. Using the above lemma, there exist linear codes for achieving the weak secrecy capacity as long as \( q \), the code alphabet size, is

\[
q \geq \left( \frac{M - 1}{\mu - 1} \right) M + \binom{n}{k} \tag{6.39}
\]

using the fact that the maximum independent edges is \( M \), and at most there are \( \binom{n}{k} \) different destination nodes. This finalizes the proof.

In this section, we stated the results for functional partial-repair. In practice, exact partial-repair is also interesting. Exact partial-repair does not require communications of the updated codes (of the new fragments) and it is also easier for data collectors to download the file when there are some systematic nodes (and remain systematic by exact partial-repair). The optimal bandwidth, and strong and weak secrecy capacities are in general unknown for exact partial-repair. Yet, the results in functional partial-repair are useful for exact partial-repair in the senses that the optimal bandwidth in functional partial-repair serves as a lower bound for the exact partial-repair, and the derived secrecy capacities in functional partial-repair serve as the upper bounds of secrecy capacities in exact partial-repair. In the next section, we show in a scenario that these bounds are tight and the bounds can be achieved for exact partial-repair.

6.4 Optimal and Secure Codes for Exact Partial-Repair

In this section, we provide an explicit code construction for the exact partial-repair that achieves the lower bound in Corollary 6.1. This implies that the proposed code is optimal in partial-repair bandwidth. Next, we construct the secure code by encoding the source file fragments.

6.4.1 Optimal Exact Partial-Repair

We first divide the source file into \( M = k^2 \) fragment\(^3\). Thus, each node stores \( M/k = k \) fragments before data loss. We construct a \( k \times k \)-matrix \( S \) which contains the source file fragments as

\[ q = 2, k = 4, \text{ the fragment size} = \left\lceil \frac{2^{20}}{4^2} \right\rceil = 2^{16} \text{ bits}. \]
\[ S = \begin{pmatrix}
  s_{11} & \cdots & s_{1k} \\
  s_{21} & \cdots & s_{2k} \\
  \vdots & \ddots & \vdots \\
  s_{k1} & \cdots & s_{kk}
\end{pmatrix}. \quad (6.40) \]

Without loss of generality, we assume that there are \( k \) nodes that store the unencoded fragments. These nodes are called systematic nodes. In addition, there are \((n-k)\) parity nodes that store coded fragments. We store in \( k \) systematic nodes the symbols in rows of matrix \( S \). That is, the \( i \)-th systematic node stores \( k \) fragments in the \( i \)-th row of matrix \( S \). Then, we store coded fragments in \( n-k \) parity nodes. To get the coded fragments in parity nodes, we construct matrix \( P \) as

\[ P = \Phi S, \quad (6.41) \]

where \( \Phi \) is a \((n-k) \times k\)-dimensional Cauchy matrix \cite{Hei95}, with elements from a finite field \( \mathbb{F}_q \), for \( q > n \). We store in parity node \( i \) the symbols in the \( i \)-th row of matrix \( P \). If \( P_i \) denotes the vector in the \( i \)-th row of matrix \( P \), then the coded fragments in parity node \( i \) are elements of vector \( P_i \) as

\[ P_i = \Phi_i S, \text{ for } i = 1, \ldots, n-k. \quad (6.42) \]

Here, \( \Phi_i \) denotes the \( i \)-th row of matrix \( \Phi \). By this construction, the code on the storage nodes is an \((n, k)\)-MDS code.

**Proposition 6.2.** The above code is an \((n, k)\)-MDS code.

**Proof.** It is straightforward to verify that, by selecting any \( k \) storage nodes the encoding vectors are independent (due to Cauchy matrix properties \cite{Hei95}) and thus the original file can be reconstructed by, e.g., Gaussian elimination method. \( \square \)

### 6.4.2 Secure Partial Repair for Systematic Nodes

Now, we describe the process of exact partial-repair. There are \( n-k \) faulty storage nodes, each of which has lost \( \xi \) fragments. Let us assume \( n-k \) faulty storage nodes are parity nodes, and thereby, \( k \) correct storage nodes are systematic nodes. Let \( P_{ij} \) denote the element in row \( i \) and column \( j \) of matrix \( P \). Without loss of generality, assume that nodes labeled as \( 1, \ldots, k \) are systematic nodes and nodes \( k+1, \ldots, n \) are parity nodes. The partial repair proceeds as the following steps:

Step 1) Systematic node \( i \) transmits fragments \( v_{(i,k+1)}^1, \ldots, v_{(i,k+1)}^u, \ldots, v_{(i,k+1)}^\xi \) to node \( k+1 \), where

\[ v_{(i,k+1)}^u = S_i b_u \text{ for } u = 1, \ldots, \xi. \quad (6.43) \]

Here, \( b_1, \ldots, b_\xi \) are rows of a \( \xi \times k \) Cauchy matrix with elements taken from \( \mathbb{F}_q \), where \( q \geq k + \xi \). This step runs for all \( i \in \{1, \ldots, k\} \).
Step 2) In node $k + 1$, first $\xi$ lost fragments in the node are recovered. Suppose that fragments $P_{1j}, \ldots, P_{\xi j}$ are lost. By the code construction, we have $P_{1j} = \Phi_1 S_{j'}^T$. For recovering the lost fragments, the node calculates

$$
\Phi_1 \begin{pmatrix}
 v_{1,(k+1)}^1 & v_{1,(k+1)}^2 & \cdots & v_{1,(k+1)}^\xi \\
v_{2,(k+1)}^1 & v_{2,(k+1)}^2 & \cdots & v_{2,(k+1)}^\xi \\
\vdots & \vdots & \ddots & \vdots \\
v_{k,(k+1)}^1 & v_{k,(k+1)}^2 & \cdots & v_{k,(k+1)}^\xi \\
\end{pmatrix} = Z
$$

and obtains
\[ \Phi_1 Z = \left( \Phi_1 S_{j_1}^T \cdots \Phi_1 S_{j_\xi}^T \right) \begin{pmatrix} b_{1,j_1} & \cdots & b_{1,j_\xi} \\ \vdots & \ddots & \vdots \\ b_{\xi,j_1} & \cdots & b_{\xi,j_\xi} \end{pmatrix} \]

the desired terms

\[ + \left( \Phi_1 S_{r_1}^T \cdots \Phi_1 S_{r_{k-\xi}}^T \right)_{r_j \notin \{j_1, \ldots, j_\xi\}} \begin{pmatrix} b_{j_1,i_1} & \cdots & b_{j_\xi,i_1} \\ \vdots & \ddots & \vdots \\ b_{j_1,i_\xi} & \cdots & b_{j_\xi,i_\xi} \end{pmatrix}. \tag{6.45} \]

the interfering terms

Node \( k + 1 \) cancels the interfering term in \((6.45)\) by using its side information. Then the lost fragments can be retrieved by using the fact that matrix \( B \) is invertible.

Step 3) In the next step, node \( k + 1 \) calculates

\[ \Phi_{t-k} Z \tag{6.46} \]

and sends \( \xi \) elements of the above vector to node \( t \). This step runs for all \( t \in \{k + 2, \ldots, n\} \).

Step 4) Suppose \( \xi \) fragments are lost in node \( t \). Parity node \( t \) recovers its lost fragments the same way as node \( k + 1 \) by removing the interfering terms. This operation is repeated for all \( t \in \{k + 2, \ldots, n\} \).

Step 5) If there is a systematic node which is faulty (lost \( \xi \) fragments), then we first change the variables such that again we have \( k \) systematic nodes and \( n - k \) parity nodes. Then we proceed through steps (1)-(4).

Proposition 6.3. The proposed code is optimal in partial-repair bandwidth.

Proof. For the recovery of the lost fragments in faulty storage nodes, in the proposed code we transmit \( k \xi \) fragments in Step 1 and \((n - k - 1)\xi \) fragments in Step 3. Thus, in total, we transmit \( k \xi + (n - k - 1)\xi = (n - 1)\xi \) fragments. Thus, the proposed code achieves the lower bound in Proposition 6.1 and it is optimal. \( \square \)

In the following, we illustrate the process by an example in a distributed storage system with parameters \( n = 6, k = 3, M = 9 \), and \( \xi = 1 \).

Example 6.2. Consider a distributed storage system in Fig. 6.5. The source file contains 9 fragments \( S = \{a_1, b_1, z_1, a_2, b_2, z_2, a_3, b_3, z_3\} \), which is coded by a \((6,3)\)-MDS code. We illustrate in the figure the process of exact partial-repair code for recovering the lost fragments in nodes 4, 5, and 6. We show how the side information can be properly exploited for the the exact partial-repair.
6.5. Discussion and Numerical Results

Next, we aim to make the system strongly secure. We use coset codes [ERS07] as outer codes in the above distributed storage system. Then, we show that it can provide strong security when an eavesdropper overhears repairing fragments in partial repair.

In the distributed storage systems studied in this section, assume a worst case that the eavesdropper overhears all the fragments in a stage of partial-repair. In one stage of partial repair, the maximum independent messages that the eavesdropper can access is \( \mu = k\xi \). For that, we have \( C_{ss} \leq M - k\xi \). We provide the codes that achieve the secrecy capacity by substituting \( k\xi \) random symbol, which are taken uniformly from \( \mathbb{F}_q \), into the source matrix as follows.

\[
S = \begin{pmatrix}
s_{11} & \cdots & s_{1(k-\xi)} & z_{11} & \cdots & z_{1\xi} \\
s_{21} & \cdots & s_{2(k-\xi)} & z_{21} & \cdots & z_{2\xi} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
s_{k1} & \cdots & s_{k(k-\xi)} & z_{k1} & \cdots & z_{k\xi}
\end{pmatrix}.
\] (6.47)

The partial repair process remains the same as before.

**Proposition 6.4.** The above proposed code is strongly secure and achieves the secrecy capacity \( C_{ss} = M - k\xi \).

**Proof.** Consider a distributed storage system, where at most \( \mu = k\xi < M \) independent fragments are overheard by an eavesdropper.

We represent the source fragments by vector \( \mathbf{v} = [s_{11}, \ldots, s_{k(k-\xi)}, z_{1}, \ldots, z_{k\xi}] \), then we have

\[
H = \left( \mathbf{I}_{k(k-\xi) \times k(k-\xi)} \mathbf{0}_{k(k-\xi) \times k} \right).
\] (6.48)

If \( \mathbf{C}_w \) is the matrix of encoding vectors of \( k\xi \) fragments from the correct storage nodes to the faulty storage nodes, then by the above proposed code that Condition (6.28) is always satisfied. That is, the rank \( \begin{pmatrix} H \\ \mathbf{C}_w \end{pmatrix} = M \). This finalizes the proof. \( \square \)

We can also use coset codes as outer codes to achieve weak secrecy capacity.

6.5 Discussion and Numerical Results

We derived an upper bound of strong secrecy capacity in Corollary 2 as

\[
C_{ss} \leq \min_{D \subseteq [n], |D| = k} \sum_{i \in D} |P_i|.
\] (6.49)

This upper bound shows that the loss pattern over storage nodes can affect the secrecy capacity. In other words, when lost fragments are happened burstly over
a few number of nodes, the secrecy capacity is lower than the case where lost fragments are uniformly distributed among all nodes. In Fig. 6.6 we compare these secrecy capacity upper bounds for these two cases. This result implies a practical consideration in building distributed storage systems. A distributed storage system with all nodes having equal reliability (homogeneous system) has higher secrecy capacity than a heterogeneous distributed storage system (in which different storage nodes have different reliability).

A part of our study in Section 6.4 provided deterministic codes for achieving strong security. We can show the gain of using deterministic codes compared to using random network codes over a large field size. To show that, suppose that $\Gamma$ is the total number of repairing fragments. If we use random network codes over a large finite field size for partial repair, then with high probability these $\Gamma$ fragments will be independent. In this case the amount of data that can be securely stored in the distributed storage system is $M - \Gamma$, which is less than that the secrecy capacity achieved by our proposed coding scheme ($M - \mu$, and $\mu < \Gamma$). In Fig. 6.7 we compare the secrecy capacity of a system using random network codes (RNC) with a system using our proposed optimal codes.

### 6.6 Conclusions

We studied partial repair in distributed storage systems and the security of partial repair where an eavesdropper has access to a number of repairing fragments. We first derived strong and weak secrecy capacities for functional partial-repair. Then, we showed that the secrecy capacities can be achieved for exact partial-repair in
6.7 Appendices

6.7.1 Proof of Theorem 6.1

The proof is inspired by the method adopted in [CW+14] for finding the optimal solution in a data exchange problem. We prove by first formulating the partial repair problem into an instance of single-source network coding problem in a multicast network and then using the network coding results in the network information flow [ACLY00].

There are two rounds of communication for multiple faulty node partial-repair in a fully-connected network. We construct the network-coding graph \( \mathcal{G} = (\mathcal{N}, \mathcal{A}) \), for one stage of repair with a given \( \mathcal{P} = \{P_1, P_2, \ldots, P_n\} \). The set \( \mathcal{N} \) is the set of nodes and the set \( \mathcal{A} \) is the set of edges in the graph that is constructed. For each storage node \( i \), we consider these virtual nodes as \( v^0_i, v^1_i, v^2_i, w^0_{iz}, w^1_{iz}, w^2_{iz} \), for \( z \in [n] \). Then, the set \( \mathcal{N} \) contains the the source node, denoted by \( s \), data collectors \( DC_1, \ldots, DC^{\frac{n}{k}} \), and \( v^0_i, v^1_i, v^2_i, w^0_{iz}, w^1_{iz}, w^2_{iz} \), for \( i, z \in [n], i \neq z \) and where node \( z \) is a faulty node.

The set \( \mathcal{A} \) in the graph is constructed as the following steps:

- For each \( i \in [n] \), there is an edge of capacity \( |P_i| \) from \( s \) to \( v^0_i \).
- There is an edge of infinite capacity from \( v^0_i \) to \( v^1_i \), for \( i \in [n] \).

![Figure 6.7: Comparing random network coding with our deterministic optimal codes.](image-url)
There is an edge of infinite capacity from $v_i^1$ to $v_i^2$, for $i \in [n]$.

There is an edge of capacity $\beta_{ij}$ from $v_i^0$ to $w_j^0$, for $i \in \{1, \ldots, r\}$ and $j \in \{r+1, \ldots, n\}$. These edges represent the information sent from correct storage nodes to faulty nodes in first round of partial repair process.

There is an edge of capacity $\beta_{ij}$ from $v_i^1$ to $w_j^2$, for $j \in \{r + 1, \ldots, n\}$ and $i \in \{r + 1, \ldots, n\}$. These edges represent the information sent from faulty storage nodes to other faulty nodes in second round of partial repair process.

There is an edge of infinite capacity from $w_i^1$ to $v_i^1$, for $i \in \{1, \ldots, n\}$.

There is an edge of infinite capacity from $w_i^2$ to $v_i^2$, for $i \in \{1, \ldots, n\}$.

There is a data-collector which connects by $k$ infinite capacity edges to a set of $k$ arbitrary nodes among $\{v_1^2, \ldots, v_n^2\}$.

As an example the information flow graph in partial repair problem in Example 1 is given in Fig. 6.8.

If $\beta_{ij}$'s are chosen such that all the cut set bounds on the graph $G$ are satisfied, then there would be a network coding solution for the multicast of $M$ fragments.
from $s$ to $\{DC_1, \ldots, DC_{\binom{n}{k}}\}$. This network coding solution also solves the partial repair problem. What remains to be shown is that the inequalities defining $R(P)$ are satisfied if and only if any cut separating $s$ from $DC_r$ in $G$ has capacity at least $M$. To this end, suppose we have a cut $(S, S^c)$ satisfying $s \in S^c$ and $DC_j \in S$ for $r \in [\binom{n}{k}]$. We will modify the cut $(S, S^c)$ to produce new cut $(S', S'^c)$ with capacity less than or equal to the capacity of the original cut $(S, S^c)$.

Consider a data collector which is connected to storage nodes $i_1, \ldots, i_k$. Let $DC_r \in S$ and $s \in S^c$. Initially let $S' = S$. Modify the cut $(S', S'^c)$ as follows:

- M1) If node $i$ is inside the set $i_1, \ldots, i_k$, then place node $v^0_i, v^1_i, v^2_i$ in $S'$.
- M2) Otherwise, place node $i$ in $S'^c$.

Further, we modify the cut as follows:

- M3) If node $i$ is a faulty node, and node $j$ is also a faulty node inside the set $i_1, \ldots, i_k$, then place node $w_{ij}$ in $S'$.
- M4) Otherwise, place node $w_{ij}$ in $S'^c$.
- M5) If node $i$ is a correct storage node, and node $j$ is a faulty node inside the set $i_1, \ldots, i_k$, then place node $w_{ij}$ in $S'$.
- M6) Otherwise, place node $w_{ij}$ in $S'^c$.

We can calculate the capacity of the modified cut $(S', S'^c)$, and see that its capacity is greater than or equal to $M$ if and only if

$$\sum_{j \in S'} \sum_{i \in S'^c} \beta_{ij} \geq M - |\cup_{j \in S'} P_j|$$

Thus, any modification of the cut cannot have a capacity greater than the capacity of the cut. Hence, the cut $(S, S^c)$ also has capacity greater than or equal to $M$ if the above inequality is satisfied. In Fig. 6.9, we illustrate a cut $(S, S^c)$ and its modified minimal cut $(S', S'^c)$ for the graph $G$ corresponding the partial repair in Example 1.
Figure 6.9: Cut analysis in information flow graph for the multiple faulty node partial repair in Example 1
Part IV

Consistent Distributed Storage systems
Chapter 8

Conclusions

8.1 Concluding Remarks

This thesis studied the problems and challenges in distributed storage systems and the role of coding in these systems from different aspects such as reliability, availability, security, consistency, robustness against network partitioning (in particular node and link failure), and communication costs (in particular bandwidth and transmission costs). We studied distributed storage systems in multi-hop networks and considered the transmission cost in repair. In addition, we studied the case of distributed storage systems where the links between storage nodes are lossy links and the transmitted data between storage nodes might be lost. Next, we introduced the concept of partial node failure and then we derived the optimal bandwidth-storage tradeoff for partial repair. Moreover, we studied data security in distributed storage systems. We also studied consistency in data read operations in distributed storage systems.

In Chapter 3, we proposed surviving nodes cooperation in the repair process. When surviving nodes cooperation is allowed in the system, the intermediate storage node can combine their received data with their own stored data and then transmit encoded data towards the new node. In these systems, we formulated an optimization problem to find the optimal-cost in repair. We showed that surviving nodes cooperation can reduce the repair-cost. We derived the required code alphabet size for achieving the optimal cost for functional repair. For exact repair, we proposed explicit code construction in tandem and grid networks. The study showed that in some network topologies increasing the number of surviving nodes in repair is not beneficial in reducing the repair costs.

In Chapter 4, we studied the repair problem in packet erasure networks. We derived the fundamental bandwidth-storage tradeoffs for various packet erasure probabilities. We showed that the fundamental tradeoff can be achieved when the file size is infinitely large. For a finite file size, we studied the probability of successful repair. We defined the notion of practical repair-bandwidth, meaning the repair bandwidth for achieving a given probability of successful repair. We also proposed...
the concept of dedicated-for-repair (DR) storage nodes. We showed that DR storage nodes can reduce the repair bandwidth.

In Chapter 5, we modeled partial node failure. We studied partial repair when a part of data stored in a storage node is lost. We proposed a two-layer coding scheme and derived the fundamental bound of repair-bandwidth when the two-layer coding is used. Interestingly, we showed that two extreme points in new bandwidth-storage tradeoff can be constructed by MDS codes.

In Chapter 6, we studied partial repair where there are multiple faulty nodes (nodes that have lost parts of their stored data). We derived the optimal bandwidth in partial repair. Next we studied the security in partial repair and derived the strong and weak secrecy capacities.

In Chapter 7, we studied the role of coding in consistent distributed storage systems where the links between storage nodes may fail with a given probability. We proposed an optimal encoding of multi versions of a file such that the probability of reconstruction is maximized.

8.2 Future Work

There are several possible directions to extend the results of this thesis. Here, we state a number of suggestions for future work, as follows.

Surviving Node Cooperation using non-MDS Codes We studied the optimal-cost repair problem using surviving node cooperation in multi-hop networks. In our study the source file is encoded by MDS codes. This study can be extended to more general codes than MDS codes. We expect that increasing individual nodes storage space can reduce the repair-cost. Then one can investigate the fundamental tradeoff between repair-cost and storage space for different network topologies.

The Fundamental Bandwidth-Storage Tradeoff in Fully Connected Networks In our study, we found that it is difficult to derive a closed-form relation between repair-cost and storage in general network topologies. Since, fully connected network is an applicable model for data-centers, it is interesting to focus on fully-connected networks and derive the optimal cost-storage tradeoff. At starting, we suggest deriving the fundamental bandwidth-storage tradeoff in fully connected networks, i.e., assuming that all the links has equal transmission costs. Next, one can introduce different costs for the use of different links.

Exact Repair using Surviving Node Cooperation In our study, we derived the optimal bound for functional repair and then showed that exact repair is achievable in some scenarios, e.g., in tandem and grid networks. The study on the exact repair using surviving node cooperation in general network topologies is interesting. Surviving nodes cooperation may offer smaller repair-cost, smaller repair-bandwidth
or simpler code construction for exact repair.

**DR Storage Nodes for Exact Repair** In our study, we showed the gain of DR storage nodes for functional repair. The use of dedicated-for-repair storage nodes can also be studied for exact repair. We expect that DR storage nodes can provide lower repair-bandwidth or simpler code construction for exact repair.

**Weighted Locally Repairable Codes** Erasure codes for partial repair can be extended to weighted locally repairable codes. We proposed the idea of partial repair. In partial repair, there are a group of nodes that communication between them does not have any cost, yet there is a cost for communication to an external node. Thus, partial repair can be considered as two-weight locally repairable codes. This can be extended to more general real-valued weights in locally repairable codes.

**Security in Consistent Distributed Storage Systems** In this thesis, we observed that in order to have a consistent system, storage nodes exchange information and update their stored data. If an eavesdropper overhears the exchanged information, or if the eavesdropper has access to a number of storage nodes, it may obtain some information about the stored files. Thus, it is important to study security in consistent distributed storage systems.

**Computing, Storage, and Communication Tradeoffs** Studying the fundamental tradeoffs on the triangle of computing, storage, and communication in distributed computing systems is interesting and can be studied as future work. In the literature it has been studied that by increasing the computation in storage nodes, the communication costs can be reduced. In another direction of research, it has been shown that by increasing the storage space, the communication between storage can be reduced. However, the interrelation between storage, computation and communication has not been fully studied.


Serveh Shalmashi. Cooperative spectrum sharing and device-to-device communications. 2014.


