Title
Patterned Erasure Correcting Codes for improved Storage and Communication Efficiency in Blockchain Systems

Author
Mitra, Debarnab

Publication Date
2020

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Patterned Erasure Correcting Codes for improved Storage and Communication Efficiency in Blockchain Systems

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Electrical and Computer Engineering

by

Debarnab Mitra

2020

ABSTRACT OF THE THESIS

Patterned Erasure Correcting Codes
for improved Storage and Communication Efficiency
in Blockchain Systems

by

Debarnab Mitra
Master of Science in Electrical and Computer Engineering
University of California, Los Angeles, 2020
Professor Lara Dolecek, Chair

Blockchains are decentralized ledgers which store the sequence of transactions in the form of a hash chain. However, this decentralization requires each node in the network to store the entire blockchain, an operation which incurs significant storage costs. Erasure coding and network coding techniques were previously introduced to mitigate this storage burden. In this thesis, we introduce a technique that leverages the patterned nature of node failures in blockchain systems to design a coding scheme called PARE (Pattern Aware Redundancy for Erasures), which minimally corrects only a predefined set of node failure patterns (called a patterned set), in the sense that the code guarantees to correct only the erasures present in the patterned set and gives no guarantees about erasure patterns that are not present in the patterned set. PARE is able to significantly reduce storage costs in blockchain systems compared to previous erasure coding techniques. We then modify PARE to a locally recoverable coding scheme called PARE-LRC which corrects all single node failures locally while still minimally correcting only a predefined set of node failure patterns. In situations where

single node failures are more prone to occur, this approach lowers communication cost and provides a better trade off between communication cost and storage cost compared to other techniques used for blockchain systems.

The thesis of Debarnab Mitra is approved.

Danijela Cabric

Lieven Vandenberghe

Lara Dolecek, Committee Chair

University of California, Los Angeles

2020

To my parents, Gopa and Debashis

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor Professor Lara Dolecek for giving me the opportunity to work under her guidance at the Laboratory of Robust Information Systems at UCLA. This thesis would not have been possible without her support and encouragement to which I shall always remain indebted to. Through our interactions, I have learned many valuable lessons about conducting research which will remain with me forever. I am grateful to all my committee members, Professor Dolecek, Professor Vandenberghe and Professor Cabric whose classes I thoroughly enjoyed as a student, which truly inspired me to carry out research. I thank them for their time devoted serving on my committee.

I would like to thank UCLA for the diverse curriculum that it offers that allowed me to explore different subjects and learn new concepts and ideas that proved helpful in my research. It also provided me the opportunity to interact with many graduate and undergraduate students as a Teaching Assistant that helped me broaden my perspective about engineering. I would also like to thank Ryo Arreola and Deeona Columbia for their help during the course of my masters.

I would also like to acknowledge the members of the LORIS lab at UCLA with whom I have many fruitful discussions, Siyi Yang, Zehui Chen, Lev Tauz, Ruiyi Wu and Dr. Homa Esfahanizadeh. They have truly guided and supported me through my research work.

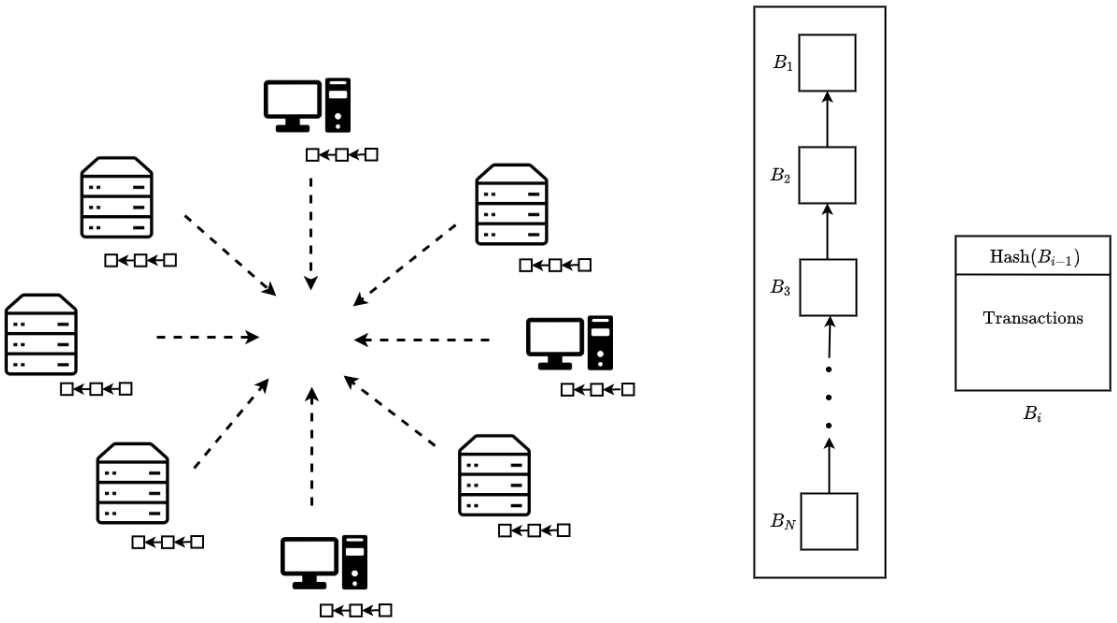
The Research is supported in part by the Guru Krupa Scholarship and NSF-BSF grant no. 1718389.

CHAPTER 1

Introduction

Blockchains provide a method to maintain a distributed ledger of transaction data, and form the backbone of various cryptocurrencies like Bitcoin and Ethereum. The decentralized nature of blockchains provides a trust-free setting and avoids the need of any central authorities thereby also improving the security of a system. This property has in turn led to the expansion of blockchain applications outside cryptocurrencies across diverse fields, including industrial IoT [BM16], [TR17], healthcare [Met16], medicine [AEV16], supply chain management [CW17], and government services [Swa]. Decentralization, however, implies that each node in the network stores the entire ledger of transactions, which incurs significant storage costs as the size of the blockchain increases. For example, the size of the Bitcoin blockchain has reached over 275GB as of May 2020 [Bit20] despite its low throughput, and the size of high throughput blockchains like Ripple has grown over 9TB [Rip19].

Currently, solutions to reduce storage costs can be classified into two categories [SR19]: (i) running *light* clients as in simplified payment verification (SPV) where only the elementary information of each block is stored, (ii) *block prunning* which involves storing the most recent blocks and deleting old blocks. These solutions reduce storage costs at the expense of losing information. Blockchain systems can be conveniently viewed as distributed storage systems with *full replication*. To reduce storage costs, erasure coding techniques for blockchain systems have been proposed in [DZW18], [al18a], [Wil14], [RV18]. These solutions use the concept of *coded sharding*, where the blockchain is partitioned into k shards, and n coded shards are generated from these k partitions using an (n, k) maximum dis-



(a) Peer to peer network in Blockchain

(b) Blockchain Ledger

Figure 1.1: Blockchain System

tance separable (MDS) code. Each node stores one coded shard thus reducing the storage at each node to $\frac{1}{k}$ times the original storage. In this thesis we show that it is possible to reduce storage costs even further by leveraging the patterned nature of erasures present in the blockchain system.

1.1 Primer on Blockchains

Blockchain is a tamper proof ledger that enables transaction verification in an untrusted environment by enabling the untrusted parties to come to a distributed consensus without any central authority. This is implemented by a peer to peer network of nodes, each maintaining its own copy of the ledger. The ledger is in the form of a hash chain of blocks of transactions where each block has within, the hash of the previous block in the ledger. Fig. 1.1a illustrates the peer to peer network in a blockchain system consisting of many nodes

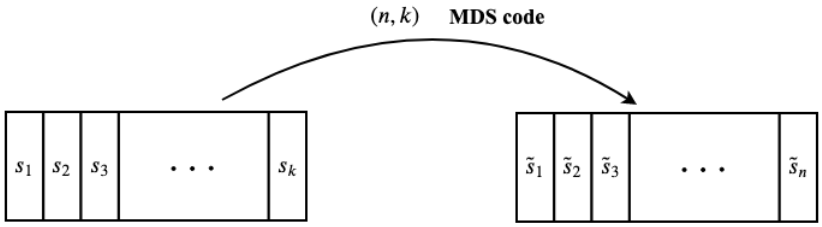


Figure 1.2: Coded Sharing of the blockchain ledger

each storing a copy of the blockchain ledger. The blockchain legder that is stored at each node is illustrated in Fig. 1.1b. It consists of blocks of transactions $B_1, B_2, \ldots, B_N$, where block B_i stores the hash of block B_{i-1}. If transactions in a block B_i are tampered, its true hash gets altered and hence the subsequent block B_{i+1} become invalid as it is no longer storing the correct hash of its previous block. Thus to preserver the validity of the ledger, all the subsequent blocks have to be altered to make them store the correct hash which has high computational complexities as the block gets deeper into the ledger. A transactions posted by any node in the network is broadcast to all the other nodes in the network, who then verify the transaction and then add it to the hash chain. Consensus about the correct state of the chain is achieved by taking a majority vote among all the nodes in the network.

Thus as described earlier, the blockchain system requires each node to store the entire blockchain ledger which incurs significant storage costs as more and more transactions are made in the system. To reduce this storage room requirement, coded sharding techniques were proposed in [DZW18], [al18a]. As mentioned earlier, in the coded sharding technique (for a blockchain with n nodes), the blockchain ledger is partitioned into k partitions $s_1, s_2, \ldots, s_k$ and an (n, k) MDS code is used to generate n coded shards $\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_k$. Each node stores only one coded shard thus reducing the storage requirement at each node by a fraction k. This is illustrated in Fig. 1.2. As long as k valid coded shards are present in the peer to peer network, the correct state of the blockchain can be recovered due to the property of MDS codes.

1.2 Contributions

A key challenge in reducing storage costs in blockchain systems is the increased recovery communication cost which is the average number of nodes accessed to recover the lost data. When a single node fails, traditional blockchain systems can recover this failure by accessing the information stored at any other node since each node stores a copy of the entire blockchain ledger. Although coded sharding techniques are able to reduce storage cost in a blockchain system to $\frac{1}{k}$ times the original storage, the operation comes at the expense of increased communication cost to k times the original cost; in order to recover a node failure in coded sharding, k nodes have to be accessed owing to the property of MDS codes. Thus there is trade off between storage and communication cost in blockchain systems. In particular, we study the following performance measures of blockchain systems: *storage efficiency:* - measured as the ratio of the total blockchain size and the average storage at each node, *communication cost:* - measured as the average number of nodes accessed to recover the lost data when node failures occur. The goal of this work is to design coding schemes which have good performance with respect to both measures and also have the best possible trade off between them.

Data contracts in certain blockchains impose an expectation on the uptime of nodes allowing each node to go down periodically [Wil14]. Nodes that have dedicated farming hardware have more uptime compared to smaller nodes who farm data on their laptops. Thus, different nodes have different uptimes, and hence different periodicity of failures. When nodes fail with different periodicity, only some specific patterns of detrimental erasures are possible. Given that only certain patterns of node erasures can occur, traditional erasure codes that are designed to correct all possible erasure patterns are not optimal in terms of maximizing storage efficiency in blockchain systems. In this thesis, we design a coding scheme called PARE (Pattern Aware Redundancy for Erasures) that corrects only these specific erasure patterns; we refer to the correction of only specific erasure patterns without

guarantees on the remaining erasure patterns as *minimal correction*. By doing so, PARE significantly increases the storage efficiency compared to other schemes. Patterned nature of node failure also enables us to achieve a better trade off between storage efficiency and communication cost compared to coded sharding blockchain systems. We do so by designing a coding scheme called PARE-LRC that can locally correct all single node failures and minimally correct only the specific patterns of node erasures.

A drawback of using coding methods to store blockchain data is the issue of constant redesign. In methods like coded sharding, the (n, k) code used depends on the number of nodes currently in the system. If a new node is added or if it leaves the system, a new (n', k') code (where n, n' and k, k' are not necessarily the same) has to be designed and the coded shards stored at all the existing nodes have to be regenerated. This case leads to an increased system complexity. We show that for the PARE-codes, in certain situations, this redesign complexity is significantly lower compared to coded sharding.

1.3 Organization

The rest of the thesis is organized as follows. In Chapter 2, we introduce preliminaries were we describe the patterned erasure set model that we use through the thesis. In Chapter 3, we propose our PARE coding scheme and show that it is optimal in terms of maximizing storage efficiency. In Chapter 4, we propose PARE-LRC codes that improve upon the trade off between storage efficiency and communication cost although they have lower storage efficiency compared to PARE-codes designed in Chapter 3. In Chapter 5, we address the redesign complexity of our proposed codes. Our simulation results are given in Chapter 6. Finally, the thesis is concluded in Chapter 7 with a short discussion.

CHAPTER 2

Blockchain System Model

2.1 Patterned erasure model

Let the blockchain system with a full ledger size of B have n nodes $\{N_1, N_2, \ldots, N_n\}$ and let $\mathcal{P} = \{P_1, P_2, \ldots, P_{|\mathcal{P}|}\}$ be a set of $|\mathcal{P}|$ different patterns of node failures that can take place, where each P_i (called a patterned erasure set) is a subset of $\{N_1, N_2, \ldots, N_n\}$ indicating a set of nodes that can fail together. We assume that a subset of failures of each patterned set P_i is also possible. We denote the maximum and minimum size subsets in $\mathcal{P}$ by $t_{max}(\mathcal{P})$ and $t_{min}(\mathcal{P})$ respectively, i.e., $t_{max}(\mathcal{P}) = \max|P_j|$, and $t_{min}(\mathcal{P}) = \min|P_j|$, $1 \leq j \leq |\mathcal{P}|$. We make the assumption that the all-node failure is not possible i.e., $t_{max}(\mathcal{P}) < n$. At any given instance, let $\mathcal{I}$ denote the indices of all nodes in an erasure and let $N_{\mathcal{I}} = \{N_i | i \in \mathcal{I}\}$ denoting the set of nodes in the erased state. Our coding technique only requires that at any instance the set of nodes failing $N_{\mathcal{I}}$ is a subset of one of the patterned erasure sets in $\mathcal{P}$, i.e., $N_{\mathcal{I}} \subseteq P_i$ for some $i \in \{1, 2, \ldots, |\mathcal{P}|\}$. For ease of exploration, we assume that the underlying cause of these patterned erasure set $\mathcal{P}$ has a periodic nature of node failures. However, these techniques can also be used in any other scenarios that have a patterned nature of node failures which do not necessarily arise due to periodic node failures. Formally, we consider the following definitions.

Definition 1. *$N_{\mathcal{I}}$ is said to be $\mathcal{P}$- patterned if $N_{\mathcal{I}} \subseteq P_i$ for some $i \in \{1, 2, \ldots, |\mathcal{P}|\}$.*

Definition 2. *A code $\mathcal{C}$ is called $\mathcal{P}$-correcting if it can correct all $\mathcal{P}$-patterned erasures but gives no guarantees about correcting erasure patterns that are not $\mathcal{P}$-patterned.*

A blockchain system we consider can only have $\mathcal{P}$-*patterned* erasures and we seek to design $\mathcal{P}$-*correcting* codes with good storage efficiency, communication cost and redesign complexity for such a system.

2.2 Model for periodic node failures

Here we state the model of periodic node failures that results in patterned erasures. As pointed out earlier, our work applies to any other setting where the erasures in the system can be modeled as $\mathcal{P}$-*patterned* which may or may not arise from periodic node failures.

For each node N_i in the blockchain system, we associate a (u_i, d_i, ϕ_i) tuple, where u_i is the uptime and denotes the number of time slots the node is active before failing, d_i is the downtime and denotes the number of time slots where the node can possibly fail before becoming active again, and ϕ_i is the phase and denotes the initial number of time slots the node is active before an instance of downtime. To allow for subset failures, we assume that at an instance of downtime, a node can fail with probability p. It should be noted that each $\phi_i \in [0, u_i]$. We assume a finite set of uptimes-downtime pairs $(U, D) = [(u_1, d_1), (u_2, d_2), \ldots, (u_e, d_e)]$ and each node N_i is uniformly randomly assigned a (u_i, d_i) pair from (U, D). Also, each node is randomly assigned a $\phi_i \in [0, u_i]$. Now, based on these periodicities, patterned set $\mathcal{P} = \{P_1, P_2, \ldots, P_{|\mathcal{P}|}\}$ is derived assuming that a node always fails at an instance of downtime (which is then used in the PARE and PARE-LRC coding approaches). Thus, in the blockchain system, the patterned erasures that occur will always be a subset of one of the specified patterns. We make the assumption that all nodes have (u_i, d_i) pairs such that there does not exist any time slot where all nodes fail at the same time i.e., $t_{max}(\mathcal{P}) < n$.

2.3 Performance metrics

For a coding scheme $\mathcal{C}$, we are interested in the following performance metrics:

1. *Storage efficiency:* Denoted by $\Gamma_{\mathcal{C}}$, it is defined as the ratio between the size of the entire blockchain ledger and the average storage per node, i.e.,

$$\Gamma_{\mathcal{C}} = \frac{nB}{\sum_{i=1}^{n} B_i}, \tag{2.1}$$

   where B_i is the storage size at node N_i.

2. *Communication cost:* We measure the communication cost of the system by taking into account random $\mathcal{P}$-*patterned* erasures and then calculating the average number of nodes accessed to recover the lost data when random $\mathcal{P}$-*patterned* erasures occur. Denoted by $\Lambda_{\mathcal{C}}$, communication cost is formally defined as

$$\Lambda_{\mathcal{C}} = \mathrm{E}_{N_{\mathcal{I}}}[\Lambda_{\mathcal{C}}(N_{\mathcal{I}})], \tag{2.2}$$

   where the expectation is over all $\mathcal{P}$-*patterned* erasures $N_{\mathcal{I}}$, which may have some (possibly non-uniform) distribution, and $\Lambda_{\mathcal{C}}(N_{\mathcal{I}})$ is the communication cost associated with $N_{\mathcal{I}}$.

3. *Redesign complexity:* Coded blockchain systems require redesign when nodes leave or are added to the blockchain system. We measure the redesign complexity by the probability of this redesign.

As mentioned earlier, there is a fundamental trade-off between storage efficiency and communication cost in the sense that more storage efficiency can be achieved at the expense of high communication cost. For example, *full replication* blockchain systems have low storage efficiency of $\Gamma_{\mathcal{C}} = 1$ (no storage savings) but also have low communication cost, $\Lambda_{\mathcal{C}} = 1$, since the lost information after any ($\mathcal{P}$-*patterned*) erasure can be recovered by accessing any of the active nodes as each of them is storing the full blockchain. If coded

sharding is used to correct all $\mathcal{P}$-*patterned* erasures, to obtain a $\Gamma_{\mathcal{C}} = k$, we need $\Lambda_{\mathcal{C}} = k$ (as we shortly verify); these examples demonstrate a trade-off between the two quantities. Our main goals are to design $\mathcal{P}$-*correcting* codes that have the maximum storage efficiency, and to design $\mathcal{P}$-*correcting* codes that achieve the best possible trade off between $\Lambda_{\mathcal{C}}$ and $\Gamma_{\mathcal{C}}$, while maintaining good *redesign complexity* performance.

2.4 Storage efficiency of Coded Sharding

Coded sharding is the most commonly used technique to increase storage efficiency in blockchain systems [DZW18], [RV18], [al18b]. In this technique, a blockchain with n nodes is partitioned into $k \leq n$ fragments (called shards) and n coded shards of the same size are generated using an (n, k) MDS code. Each node is made to store one coded shard there by reducing the storage cost by a factor k implying $\Gamma_S = k$. Since the method makes use of an (n, k) MDS code, it by design can correct all $n - k$ node erasure patterns. The following lemma characterizes the $\mathcal{P}$-*correcting* property of the coded sharding method.

Lemma 1. *For a blockchain with n nodes having erasure patterned set $\mathcal{P}$ and maximum erasures $t_{max}(\mathcal{P})$, a code desgined using coded sharding is $\mathcal{P}$-correcting if and only if the (n, k) MDS code used satisfies $k \leq n - t_{max}(\mathcal{P})$.*

Proof. Let the (n, k) code used in coded sharding be defined over GF(q). In order to correct all erasure patterns in $\mathcal{P}$, it is sufficient for n and k to satisfy $n - k \geq t_{max}(\mathcal{P})$, as $t_{max}(\mathcal{P})$ is the maximum number of node failures that can occur in $\mathcal{P}$. This condition is also necessary as any two codewords in the code must differ in the $n - t_{max}(\mathcal{P})$ positions where the $t_{max}(\mathcal{P})$ erasures do not take place, which puts a bound on the total number of code words, $q^k \leq q^{(n - t_{max}(\mathcal{P}))}$. □

Corollary 1. *For a blockchain with n nodes having erasure patterned set $\mathcal{P}$ and maximum erasures $t_{max}(\mathcal{P})$, the maximum storage efficiency that can be achieved using the coded shard-*

ing method is $\Gamma_{\mathcal{C}} = n - t_{max}(\mathcal{P})$*. Moreover the maximum storage efficiency is achieved using an* $(n, n - t_{max}(\mathcal{P}))$ *MDS code that corrects all* $t_{max}(\mathcal{P})$ *erasure patterns.*

The above corollary suggests that in coded sharding, it is good enough to use an $(n, n - t_{max}(\mathcal{P}))$ MDS code which corrects all $t_{max}(\mathcal{P})$ erasures patterns and there is no additional benefit in knowing the patterned nature of erasures. Lemma 1 also suggests that size of the Galois Field does not matter in improving the range of feasible k (which in turn will improve the storage efficiency), and it suffices to consider a large enough field. It is however worth noting that in coded sharding all the nodes store the same amount of the blockchain, i.e., they store the same number of shards. In the next chapter, we will see that relaxing this condition can in fact give a better storage efficiency.

2.5 Discussion

In Chapter 2 we propose the patterned erasure set model for the blockchain system. We also define the three performance metrics that we intend to optimize by leveraging the structure provided by this patterned erasure set model. We show that the coded sharding technique is unable to exploit this added structure of the partnered nature of erasures and the best storage efficiency is achieved by a worst case design of correcting the maximum number of erasures possible due to the patterned erasure set. In Chapter 3 we will design codes called PARE that have a better storage efficiency compared to coded sharding. In fact we show that of all $\mathcal{P}$-correcting codes, PARE-codes have the maximum storage efficiency.

CHAPTER 3

PARE: Coding for Maximum Storage Efficiency

In this chapter, we design a $\mathcal{P}$-*correcting* coding scheme called PARE (Pattern Aware Redundancy for Erasures) which leverages the $\mathcal{P}$-patterned nature of erasures in blockchains to improve the system storage efficiency compared to coded sharding. The coding scheme is designed by recognizing that when only $\mathcal{P}$-*patterned* erasures occur, the nodes are no longer symmetric and hence different nodes may store different number of shards in order to improve storage efficiency. In particular we have the following definition of a PARE-code:

Definition 3. *For a blockchain system with n nodes with patterned set $\mathcal{P}$, an $(n, k, \mathrm{x}, \mathcal{P})$ PARE-code is a $\mathcal{P}$-*correcting *code that partitions the blockchain into k shards and stores x_i shards at node N_i, $1 \leq i \leq n$, where $\mathrm{x} = \{x_1, x_2, \ldots, x_n\}$.*

According to the above definition, we see that coded sharding with $k \leq n - t_{max}(\mathcal{P})$ is a special case of PARE-codes where each node is restricted to store only one shard, i.e., $x_i = 1 \ \forall \ 1 \leq i \leq n$. Next, we present our construction technique to find suitable choice for (x, k) that gives us $(n, k, \mathrm{x}, \mathcal{P})$ PARE-codes that have maximum storage efficiency for all $\mathcal{P}$-*correcting* codes.

3.1 Construction of PARE Codes

Code Construction 1. ***(PARE)*** *For a blockchain of size B with n nodes having erasure patterned set $\mathcal{P} = \{P_1, P_2, \ldots, P_{|\mathcal{P}|}\}$, let $\bar{P}_j$ denote the set of nodes not in P_j. Let the blockchain be partitioned into k shards $s_1, s_2, \ldots, s_k$. Our code construction involves the*

following steps:

1. *Solve the following integer optimization problem:*

$$\begin{aligned} &\min_{x_1,\ldots,x_n,k} \quad \frac{B}{n}\frac{\sum_{i=1}^{n} x_i}{k} \\ s.t &\sum_{i:N_i\in\bar{P}_j} x_i \geq k, j = 1,2,\ldots,|\mathcal{P}| \\ &x_i \leq k, i = 1,2,\ldots,n \\ &x_i \in \mathrm{Z}^{+}, i = 1,2,\ldots,n \\ &k \in \mathrm{Z}^{++}, \end{aligned} \tag{3.1}$$

   where Z^{+} *and* Z^{++} *denote the set of non-negative and positive integers respectively.*

2. *Let* $\mathrm{x}^* = \{x_1^*, x_2^*, \ldots, x_n^*\}$ *and* k^* *be an optimal solution of* (3.1)*. Partition the blockchain into* k^* *shards* $s_1, s_2, \ldots, s_{k^*}$*. For each node* N_i*, generate* x_i^* *coded shards using linear combinations of* $s_1, s_2, \ldots, s_{k^*}$ *as follows: the* m^{th} *linear combination at node* N_i*, where* $1 \leq m \leq x_i^*$ *is* $\alpha_{i,m}^1 s_1 + \alpha_{i,m}^2 s_2 + \ldots + \alpha_{i,m}^{k^*} s_{k^*}$*, where* $\alpha_{i,m}^{\nu}$ *are from a sufficiently large finite field. Parameters* $\alpha_{i,m}^{\nu}$*'s are chosen in such a way that for each patterned set* $P_j \in \mathcal{P}$*, the matrix* M_j *formed with rows* $(\alpha_{i,m}^1 \alpha_{i,m}^2 \ldots \alpha_{i,m}^{k^*})$*,* $\forall i$ *s.t.* $N_i \in \bar{P}_j$ *and* $x_i^* \neq 0$*,* $1 \leq m \leq x_i^*$ *has rank* k^**.*

The above code construction first solves the integer optimization problem (3.1) that gives the number of blockchain partitions k^* and the number of coded shards x_i^* that need to be stored at each node. We see in the following lemma that step 2) of the procedure generates these coded shards in such a way that the code is $\mathcal{P}$*-correcting*.

Lemma 2. *Step 2) of Construction 1 is always possible. Moreover Construction 1 forms a valid* $(n, k^*, \mathrm{x}^*, \mathcal{P})$ *PARE-code.*

Proof. The constraints in (3.1) ensure that for any erasure pattern in $\mathcal{P}$, at least k^* coded shards are available among the nodes that are active after $\mathcal{P}$*-patterned* node failures occur.

This ensures that the matrices M_j in step 2) of Construction 1 have at least k^* rows. We can always choose a sufficiently large field such that each of these coefficient vectors (rows of M_j) are linearly independent and the matrices M_j have rank k^*. For example, we can choose $\alpha_{i,m}^{\nu}$ such that M_j's form Vandermonde-type matrices. Thus step 2) is always possible in the sense that we can always form matrices M_j such that they have rank k^*. Now, this step will ensure that under any patterned set P_j, the system of equations governed by the matrix M_j will give a unique solution $(s_1, s_2, \ldots, s_{k^*})$. This result implies that the code is $\mathcal{P}$-*correcting*. Thus, the code constructed is a valid $(n, k^*, \mathrm{x}^*, \mathcal{P})$ PARE-code. □

Remark 1. *Code construction 1 constructs an* $(n, k^*, \mathrm{x}^*, \mathcal{P})$ *PARE-code. Instead of using the optimal solution* $\mathrm{x}^* = \{x_1^*, x_2^*, \ldots, x_n^*\}$ *and* k^* *of problem* (3.1)*, we can use any feasible solution* $\mathrm{x} = \{x_1, x_2, \ldots, x_n\}$ *and* k *of problem* (3.1) *and follow step 2) of the construction. Using a similar argument as in Lemma 2 we can show that it will give us a valid* $(n, k, \mathrm{x}, \mathcal{P})$ *PARE-code.*

Remark 2. *In Problem* (3.1)*, it is not difficult to see that the constraints* $x_i \leq k$ *are redundant and can be removed. The optimal solution* $\mathrm{x}^* = \{x_1^*, x_2^*, \ldots, x_n^*\}$ *after the removal of the constraints will always satisfy* $x_i^* \leq k^*$*.*

The next lemma characterizes the storage efficiency of codes obtained by code construction 1.

Lemma 3. *The storage efficiency of the* $(n, k^*, \mathrm{x}^*, \mathcal{P})$ *PARE-code obtained using code construction 1 satisfies* $\Gamma_{\mathcal{C}} \geq n - t_{max}(\mathcal{P})$*.*

Proof. The objective function in problem (3.1) is the average storage per node. Now $x_i = 1$ $\forall i$, $1 \leq i \leq n$ and $k = n - t_{max}(\mathcal{P})$ (i.e., coded sharding with $(n, n - t_{max}(\mathcal{P}))$ MDS code) is feasible in problem (3.1) since

$$\begin{aligned}\sum_{i:N_i\in\bar{P}_j} x_i &= \sum_{i:N_i\in\bar{P}_j} 1\\ &= n - |P_j|\\ &\geq n - \max_j |P_j|\\ &= n - t_{max}(\mathcal{P})\\ &= k.\end{aligned}$$

and has an objective value of $\frac{B}{n-t_{max}(\mathcal{P})}$. Since (x^*, k^*) is the optimal solution of problem (3.1) it has as a lower objective compared to the above feasible point. Thus

$$\frac{B}{n}\frac{\sum_{i=1}^n x_i^*}{k^*} \leq \frac{B}{n - t_{max}(\mathcal{P})}$$

Thus the storage efficiency of the $(n, k^*, \mathrm{x}^*, \mathcal{P})$ PARE-code

$$\Gamma_{\mathcal{C}} = \frac{B}{\frac{B}{n}\frac{\sum_{i=1}^n x_i^*}{k^*}} \geq n - t_{max}(\mathcal{P}).$$

□

The above lemma states that the storage efficiency of the $(n, k^*, \mathrm{x}^*, \mathcal{P})$ PARE-code that is designed using construction 1 is always greater than that of coded sharding. The next lemma provides a way to find an optimal solution of the integer optimization problem (3.1) by just solving a linear program.

Lemma 4. *Optimal solution of problem* (3.1) *can be obtained by solving the following linear programming problem:*

$$\begin{aligned}&\min_{y_1,\ldots,y_n} \sum_{i=1}^n y_i\\ s.t &\sum_{i:N_i\in\bar{P}_j} y_i \geq 1, j = 1, 2, \ldots, |\mathcal{P}| \qquad (3.2)\\ &0 \leq y_i \leq 1, i = 1, 2, \ldots, n.\end{aligned}$$

If $y^* = (y_1^*, y_2^*, \ldots, y_n^*)$ *is an optimal solution of problem* (3.2)*, then choose* k^* *such that* $k^* \times y^* = (k^* y_1^*, k^* y_2^*, \ldots, k^* y_n^*)$ *is integral i.e., has all integer entries and choose* $\mathrm{x}^* = k^* \times y^*$*. Then* (x^*, k^*) *is an optimal solution for problem* (3.1)*.*

Proof. Lemma 4 is true since the optimization problems (3.1) and (3.2) are equivalent and we can construct the optimal solution of one from the optimal solution of the other. □

Remark 3. *From Lemma 3 and 4 we get that the storage efficiency of an* $(n, k^*, \mathrm{x}^*, \mathcal{P})$ *PARE-code is* $\frac{nk^*}{\sum_{i=1}^n x_i^*} = \frac{n}{\sum_{i=1}^n y_i^*}$ *where* (x^*, k^*) *and* y^* *are the optimal solutions of problems* (3.1) *and* (3.2) *respectively.*

Theorem 1. *For a blockchain system with patterned set* $\mathcal{P}$*, of all* $\mathcal{P}$*-correcting codes, the* $(n, k^*, \mathrm{x}^*, \mathcal{P})$ *PARE-code obtained from code construction 1 has the maximum storage efficiency.*

Proof. From the above remark the storage efficiency of an $(n, k^*, \mathrm{x}^*, \mathcal{P})$ PARE-code is $\frac{n}{\sum_{i=1}^n y_i^*}$. We prove Theorem 1 by showing that the storage efficiency achieved by any $\mathcal{P}$*-correcting* code is always less than $\frac{n}{\sum_{i=1}^n y_i^*}$. For such a $\mathcal{P}$*-correcting* coding scheme, let $B_1, B_2, \ldots, B_n$ be the amounts of the blockchain stored at $N_1, N_2, \ldots, N_n$ respectively. The storage efficiency for this coding scheme is $\frac{nB}{\sum_{i=1}^n B_i}$. Since the code is $\mathcal{P}$*-correcting*, for each patterned set $P_j \in \mathcal{P}$, B_i's must satisfy $\sum_{i:N_i \in \bar{P}_j} B_i \geq B$. Since $B \geq B_i \geq 0$, we note that $\frac{B_i}{B}$ is feasible in problem (3.2) and attains an objective value of $\frac{1}{B}\sum_i^n B_i$. Thus from the minimization in problem (3.2) we have $\sum_i^n y_i^* \leq \frac{1}{B}\sum_i^n B_i$ which implies that $\frac{nB}{\sum_{i=1}^n B_i} \leq \frac{n}{\sum_{i=1}^n y_i^*}$. This shows that the $(n, k^*, \mathrm{x}^*, \mathcal{P})$ PARE-code obtained from code construction 1 has the maximum storage efficiency. □

We next provide two examples of blockchain systems and the construction method provided in construction 1.

Example 1. *Consider a blockchain system having 6 nodes* $\{N_1, N_2, N_3, N_4, N_5, N_6\}$ *and having patterned erasure set* $\mathcal{P} = \{\{N_1, N_3, N_4, N_5\}, \{N_1, N_3, N_6\}, \{N_2, N_3, N_5, N_6\},$

$\{N_1, N_2, N_4\}, \{N_4, N_6\}\}$. *Solving problem* (3.2)*, we get* $\mathrm{y}^* = (\frac{1}{2}, \frac{1}{4}, 0, \frac{1}{2}, \frac{1}{4}, \frac{3}{4})$. *From Lemma 3.2 we get* $k^* = 4$ *and* $\mathrm{x}^* = (2, 1, 0, 2, 1, 3)$. *Thus the blockchain is partitioned into 4 shards* (a, b, c, d) *and the coded shards stored at each node is designed using step 2) of construction 1. Table 3.1 shows a possible choice of coded shards stored at each node.*

N_1	N_2	N_3	N_4	N_5	N_6
a+b+c	a	-	b	a+d	b
c+d	-	-	c	-	c
-	-	-	-	-	d

Table 3.1: A possible choice of coded shards stored at $\{N_1, N_2, \ldots, N_6\}$ for Example 1.

Using Corollary 1 we get that the maximum storage efficiency for the blockchain system using coded sharding is 2 using an $(6, 2)$ MDS code. The PARE-code constructed in the above example gives a storage efficiency of 2.67 which shows that the PARE-code has a higher storage efficiency compared to state of the art coded sharding. We note that node N_3 does not store any coded shards but can participate in blockchain transactions.

Example 2. *Consider a blockchain system again having 6 nodes* $\{N_1, N_2, N_3, N_4, N_5, N_6\}$ *and having patterned erasure set* $\mathcal{P} = \{\{N_1, N_3, N_4, N_6\}, \{N_2, N_6\}, \{N_1, N_4, N_5\}, \{N_1, N_2, N_3\}, \{N_3, N_4, N_5\}\}$. *Solving problem* (3.2)*, we get* $\mathrm{y}^* = (\frac{1}{5}, \frac{2}{5}, \frac{1}{5}, 0, \frac{3}{5}, \frac{2}{5})$. *From Lemma 3.2 we get* $k^* = 5$ *and* $\mathrm{x}^* = (1, 2, 1, 0, 3, 2)$. *Thus the blockchain is partitioned into* 5 *shards* (a, b, c, d, f) *and the coded shards stored at each node are designed using step 2) of construction 1. Table 3.2 shows a possible choice of coded shards stored at each node for this scenario.*

Again using Corollary 1 we get that the maximum storage efficiency for the blockchain system using coded sharding is 2 using an $(6, 2)$ MDS code. The PARE-code constructed in the above example gives a storage efficiency of 3.33 which is higher compared to the storage efficiency of the coded sharding method.

N_1	N_2	N_3	N_4	N_5	N_6
c+d+f	a	a+c	-	b	b+d
-	d	-	-	c	a+f
-	-	-	-	f	-

Table 3.2: A possible choice of coded shards stored at $\{N_1, N_2, \ldots, N_6\}$ for Example 2.

Remark 4. *An $(n, k, \mathrm{x}, \mathcal{P})$ PARE-code can be viewed as an $(\sum_{i=1}^{n} x_i, k)$ linear code which generates $\sum_{i=1}^{n} x_i$ codeword symbols from k information symbols. Due to the* $\mathcal{P}$-patterned *nature of erasures, in the $(\sum_{i=1}^{n} x_i)$ codeword space, the erasures that occur are also patterned with maximum erasure size $\max_{j=1,2,\ldots,|\mathcal{P}|} \big(\sum_{i:N_i \in P_j} x_i \big)$. Now if the $(\sum_{i=1}^{n} x_i, k)$ linear code is restricted to be MDS, using Lemma 1, we get that the MDS code will be able to correct all these patterned erasures if and only if $k \leq \sum_{i=1}^{n} x_i - \max_{j=1,2,\ldots,|\mathcal{P}|} \big(\sum_{i:N_i \in P_j} x_i \big)$. This is exactly the same condition as the first set of constraints in problem* (3.1) *since we can rewrite the above condition as*

$$\begin{aligned} \sum_{i=1}^{n} x_i - k &\geq \max_{j=1,2,\ldots,|\mathcal{P}|} \Big(\sum_{i:N_i \in P_j} x_i \Big) \\ \implies \sum_{i=1}^{n} x_i - k &\geq \sum_{i:N_i \in P_j} x_i, j = 1, 2, \ldots, |\mathcal{P}| \\ \implies \sum_{i:N_i \in \bar{P}_j} x_i &\geq k, j = 1, 2, \ldots, |\mathcal{P}|. \end{aligned}$$

The above remark gives us another way of designing an $(n, k, \mathrm{x}, \mathcal{P})$ PARE-code using an $(\sum_{i=1}^{n} x_i, k)$ MDS code which we summarize in the next section.

3.2 Alternative construction of PARE codes

Code Construction 2. *For a blockchain of size B with n nodes having erasure patterned set $\mathcal{P} = \{P_1, P_2, \ldots, P_{|\mathcal{P}|}\}$, let the blockchain be partitioned into k shards $s_1, s_2, \ldots, s_k$. Our modified code construction involves the following steps:*

1. *Solve problem* (3.1) *and let* $\mathrm{x}^* = \{x_1^*, x_2^*, \ldots, x_n^*\}$ *and* k^* *be the optimal solution.*

2. *Design an* $(\sum_{i=1}^n x_i^*, k^*)$ *MDS code. Partition the blockchain into* k^* *shards* $s_1, s_2, \ldots, s_{k^*}$. *Generate* $(\sum_{i=1}^n x_i^*)$ *coded shards from these* k^* *shards.*

3. *Sequentially allocate these coded shards to all the nodes with node* N_i *getting* x_i^* *coded shards.*

Remark 5. *In step 2) above, before doing the sequential allocation, any permutation of the generated coded shards will not affect the* $\mathcal{P}$*-correcting nature of the code.*

We explain the above code construction using the following example.

Example 3. *Consider the blockchain system as in Example 1 with optimal solution to problem 3.1 begin* $k^* = 4$ *and* $\mathrm{x}^* = (2, 1, 0, 2, 1, 3)$. *Thus we partition the blockchain into 4 shards* (a, b, c, d) *and use an* $(9, 4)$ *Reed-Solomon code (which is MDS) to generate coded shards* $\{c_1, c_2, \ldots, c_9\}$ *as follows:*

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \\ c_9 \end{bmatrix} = \begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \alpha_1^3 \\ 1 & \alpha_2 & \alpha_2^2 & \alpha_2^3 \\ 1 & \alpha_3 & \alpha_3^2 & \alpha_3^3 \\ 1 & \alpha_4 & \alpha_4^2 & \alpha_4^3 \\ 1 & \alpha_5 & \alpha_5^2 & \alpha_5^3 \\ 1 & \alpha_6 & \alpha_6^2 & \alpha_6^3 \\ 1 & \alpha_7 & \alpha_7^2 & \alpha_7^3 \\ 1 & \alpha_8 & \alpha_8^2 & \alpha_8^3 \\ 1 & \alpha_9 & \alpha_9^2 & \alpha_9^3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

where $\{\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_9\}$ *are distinct elements of a finite field* F *of size* $|\mathrm{F}| \geq 9$. *Now these coded shards are stored at the nodes as shown in Table 3.3. The* 9 *coded shards are sequentially allocated to the nodes in accordance to* $\mathrm{x}^* = (2, 1, 0, 2, 1, 3)$ *by allocating the first* 2 *coded shards to* N_1, *next 1 to* N_2, *none to* N_3, *and so on.*

N_1	N_2	N_3	N_4	N_5	N_6
c_1	c_3	-	c_4	c_6	c_7
c_2	-	-	c_5	-	c_8
-	-	-	-	-	c_9

Table 3.3: Coded shards stored at $\{N_1, N_2, \ldots, N_6\}$ for Example 3.

Remark 6. *As in Remark 1 instead of using the optimal solution* $\mathrm{x}^* = \{x_1^*, x_2^*, \ldots, x_n^*\}$ *and* k^* *of problem* (3.1)*, we can use any feasible solution* $\mathrm{x} = \{x_1, x_2, \ldots, x_n\}$ *and* k *of problem* (3.1) *and follow step 2) and 3) of construction 2 to get a valid* $(n, k, \mathrm{x}, \mathcal{P})$ *PARE-code.*

Remark 7. *For a feasible* (x, k) *of problem* (3.1)*, an* $(\sum_{i=1}^{n} x_i, k)$ *MDS code can be used to design a valid* $(n, k, \mathrm{x}, \mathcal{P})$ *PARE-code using code construction 2. However it is not true that all valid* $(n, k, \mathrm{x}, \mathcal{P})$ *PARE-code will result in the corresponding* $(\sum_{i=1}^{n} x_i, k)$ *linear code as MDS. For instance in Example 2 the PARE-code designed has the following* $(9, 5)$ *linear code:* $(c + d + f, a, d, a + c, b, c, f, b + d, a + f)$ *which clearly is not MDS as it cannot correct all* 4 *erasure patterns. However the* $(\sum_{i=1}^{n} x_i, k)$ *linear code will still be able to correct all erasures (in the* $\sum_{i=1}^{n} x_i$ *codeword space) resulting from* $\mathcal{P}$-patterned *erasures.*

Remark 8. *We call* $(\sum_{i=1}^{n} x_i, k)$ *codes that correct all erasures in the* $\sum_{i=1}^{n} x_i$ *codeword space resulting from* $\mathcal{P}$-patterned *erasures also as* $\mathcal{P}$*-correcting. It is easy to see that we can construct an* $(n, k, \mathrm{x}, \mathcal{P})$ *PARE-code from a* $\mathcal{P}$*-correcting* $(\sum_{i=1}^{n} x_i, k)$ *code using step 3) of code construction 2.*

We call codes constructed using construction 1 and 2 as PARE-codes.

3.3 Upper and lower limits on storage at each node

PARE-codes do not limit the storage at individual nodes. It might be possible that some of the nodes are storing a large fraction of the blockchain which leads to high storage cost at

these nodes. To prevent this from happening, we limit the number of coded shards that each node can store to a fraction u of the total number of coded shards the blockchain is divided into. This is essentially achieved by an addition of the following constraint to problem (3.1):

$$x_i \leq u \cdot k, i = 1, 2, \ldots, n,$$

where $0 < u < 1$ is a system parameter whose choice depends on the maximum permissible storage at each node. PARE-codes also allow the possibility that some of the nodes do not store any coded shards. This might result in a possible centralization of the blockchain i.e., all the coded shards are stored at a small fraction of the blockchain and majority of the nodes do not store any coded shards. This situation results in low system security and defeats the purpose of having a decentralized network. Thus, we place a lower limit l of the fraction of coded shards stored at each node by adding the following constraint to problem (3.1):

$$x_i \geq l \cdot k, i = 1, 2, \ldots, n,$$

where $0 < l < 1$. In the next chapter we will see that adding a non-zero lower limit on storage makes the communication cost of PARE-codes worse compared to coded sharding. This motivates us to design PARE-LRC codes which improve upon the communication cost while maintaining good storage efficiency.

With these additional constraints, problem (3.1) is modified into the integer optimization problem shown in problem (3.3) below,

$$\begin{aligned} \min_{x_1,\ldots,x_n,k} \quad & \frac{B}{n}\frac{\sum_{i=1}^{n} x_i}{k} \\ s.t. \quad & \sum_{i:N_i \in \bar{P}_j} x_i \geq k, j = 1, 2, \ldots, |\mathcal{P}| \\ & x_i \leq k, i = 1, 2, \ldots, n \\ & l \cdot k \leq x_i \leq u \cdot k, i = 1, 2, \ldots, n \\ & x_i \in \mathrm{Z}^{+}, i = 1, 2, \ldots, n \\ & k \in \mathrm{Z}^{++}. \end{aligned} \tag{3.3}$$

Remark 9. *Similar to Lemma 4, the optimal solution to problem* (3.3) *can be obtained by solving the linear program provided in Lemma 4 with the additional constraints*

$$l \leq y_i \leq u, i = 1, 2, \ldots, n.$$

Code construction with upper and lower limits on storage essentially follows the same procedure as construction 1 and 2 but solves problem (3.3) instead of problem (3.1). The feasibility of problem (3.3) and the storage efficiency of the resultant code depend on the choice of u and l which we state below.

Lemma 5. *Problem* (3.3) *is feasible if and only if* $u \geq \frac{1}{n-t_{max}(\mathcal{P})}$.

Proof. We prove the above lemma by showing that the problem is always infeasible when $u < \frac{1}{n-t_{max}(\mathcal{P})}$ for any choice of l. We then provide an explicit feasible point when $u \geq \frac{1}{n-t_{max}(\mathcal{P})}$ to prove feasibility. When $u < \frac{1}{n-t_{max}(\mathcal{P})}$ for any (x, k) that satisfies $l \cdot k \leq x_i \leq u \cdot k, i = 1, 2, \ldots, n$ we have for each $j \in \{1, 2, \ldots, |\mathcal{P}|\}$,

$$\begin{aligned}
\sum_{i:N_i \in \bar{P}_j} x_i &\leq (\max_i x_i)|\bar{P}_j| \\
&\leq (u \cdot k)|\bar{P}_j| \\
&< \frac{k|\bar{P}_j|}{n - t_{max}(\mathcal{P})} \\
\implies \sum_{i:N_i \in \bar{P}_j} x_i &< \min_j \frac{k|\bar{P}_j|}{n - t_{max}(\mathcal{P})} \\
&= \frac{k(n - t_{max}(\mathcal{P}))}{n - t_{max}(\mathcal{P})} \\
&= k.
\end{aligned}$$

This shows that the the first set of constraints in problem (3.3) can never be satisfied along with the condition $l \cdot k \leq x_i \leq u \cdot k, i = 1, 2, \ldots, n$ when $u < \frac{1}{n-t_{max}(\mathcal{P})}$. Now assume that $u \geq \frac{1}{n-t_{max}(\mathcal{P})}$. With regards to Remark 9, if the linear program in Lemma 4 with

the additional constraints $l \leq y_i \leq u, i = 1, 2, \ldots, n$ is feasible, then problem (3.3) will be feasible. Consider the point $y_i = u, i = 1, 2, \ldots, n$. For each j

$$\begin{aligned}\sum_{i:N_i \in \bar{P}_j} y_i &= u|\bar{P}_j| \\ &\geq u \min_j |\bar{P}_j| \\ &= u(n - t_{max}(\mathcal{P})) \\ &\geq 1.\end{aligned}$$

Thus the linear program in Lemma 4 with the additional constraints $l \leq y_i \leq u, i = 1, 2, \ldots, n$ is feasible with this choice of y_i. Thus problem (3.3) is feasible when $u \geq \frac{1}{n - t_{max}(\mathcal{P})}$. □

Lemma 6. *For $u \geq \frac{1}{n - t_{max}(\mathcal{P})}$, the PARE-code constructed with u and l limits will have storage efficiency $\Gamma_{\mathcal{C}} \geq n - t_{max}(\mathcal{P})$ if and only if $l \leq \frac{1}{n - t_{max}(\mathcal{P})}$.*

Proof. When $l \leq \frac{1}{n - t_{max}(\mathcal{P})}$, $x_i = 1 \ \forall i, \ 1 \leq i \leq n$ and $k = n - t_{max}(\mathcal{P})$ will be feasible in problem (3.3) (see Lemma 3). Thus using the same argument is in Lemma 3, we get $\Gamma_{\mathcal{C}} \geq n - t_{max}(\mathcal{P})$ for the code constructed with u and l limits. Let (x^*, k^*) be the optimal solution of problem (3.3). When $l > \frac{1}{n - t_{max}(\mathcal{P})}$,

$$\sum_{i=1}^{n} x_i^* \geq n(l \cdot k^*) > \frac{nk^*}{n - t_{max}(\mathcal{P})}.$$

Thus the storage efficiency

$$\Gamma_{\mathcal{C}} = \frac{B}{\frac{B}{n}\frac{\sum_{i=1}^{n} x_i^*}{k^*}} < n - t_{max}(\mathcal{P}).$$

□

Thus combining Lemma 5 and 6, for $u \geq \frac{1}{n - t_{max}(\mathcal{P})} \geq l$, it is always possible to design PARE-codes that will have a storage efficiency greater than that of coded sharding.

3.4 Discussion

In Chapter 3 we propose two code construction techniques to design $\mathcal{P}$-correcting codes (called PARE-codes). We show that the PARE-codes designed using construction 1 and 2 are optimal in terms of storage efficiency in Theorem 1. These code construction techniques involve solving a linear program that provides the optimal number of shards that the blockchain must be partitioned into and the optimal number of coded shards that needs to be stored at each node in the system. We also show that in situations when there is an upper and a lower limit on the storage requirement at each node, it is possible to design PARE-codes with storage efficiency greater than that of coded sharding when these storage limits satisfy certain conditions. In the next chapter we will look at the communication cost associated with PARE codes and provide our design method for PARE-LRC codes which are PARE-codes with locally recoverable property.

CHAPTER 4

PARE-LRC: Improved storage and communication cost trade off

When a single node fails, traditional *full replication* systems can recover this failure by accessing the information stored at any other node since each node is storing the entire blockchain ledger. However, PARE-codes have an MDS-like property, i.e., in order to recover the blockchain from a $\mathcal{P}$-*patterned* erasure $N_{\mathcal{I}} \subseteq P_j$, all nodes in $\mathrm{A}_{\mathrm{N}_{\mathcal{I}}} = \{N_i | N_i \in \bar{P}_j, x_i \neq 0\}$ (called the set of active nodes in $\bar{P}_j$) have to be contacted. Even if just a single node from $N_{\mathcal{I}}$ fails, all nodes in $\mathrm{A}_{\mathrm{N}_{\mathcal{I}}}$ have to be accessed. In situations where single node failures are more likely than failure of multiple nodes from the pattern of erasures, reducing the communication cost for single node failures can improve the average communication cost in blockchain systems. In this chapter, we provide our PRE-LRC coding scheme which minimally corrects a given set of node erasure patterns and has an additional property that it can correct any single node failure locally, thus reducing the average communication cost. Simulation results show that PARE-LRC codes have a better trade off between storage efficiency and communication cost compared to PARE-codes and coded sharding.

4.1 Communication Cost for CS and PARE-codes

The next Lemma characterises the communication cost for coded sharding technique.

Lemma 7. *The trade-off between communication cost and storage efficiency that can be achieved by a $\mathcal{P}$-correcting code designed by coded sharding method is $\Gamma_{\mathcal{C}} = \Lambda_{\mathcal{C}}$.*

Proof. Consider a $\mathcal{P}$-*correcting* code designed by coded sharding that achieves a storage efficiency $\Gamma_S = k$. This implies coded sharding uses an (n, k) MDS code that corrects all $\mathcal{P}$-*patterned* erasures. Now from Lemma 1 an (n, k) MDS code corrects all $\mathcal{P}$-*patterned* erasures iff $k \leq n - t_{max}(\mathcal{P})$. In other words, (n, k) MDS codes with $k \leq n - t_{max}(\mathcal{P})$ are the only possible $\mathcal{P}$-*correcting* codes obtained using coded sharding. By the MDS property, all $\mathcal{P}$-*patterned* erasures can be recovered by accessing any k active nodes. This implies that $\Lambda_{\mathcal{C}} = k$. □

In coded sharding, the maximum storage efficiency that can be achieved is $\Gamma_{\mathcal{C}} = n - t_{max}(\mathcal{P}))$, using an $(n, n - t_{max}(\mathcal{P}))$ MDS code that corrects all $t_{max}(\mathcal{P})$ erasure patterns [MD19]. This implies that there is no additional storage cost saving due to the fact that the only erasures possible are $\mathcal{P}$-*patterned*. It should also be noted that the maximum storage efficiency if obtained at the maximum communication cost point, i.e $\Lambda_{\mathcal{C}} = n - t_{max}(\mathcal{P})$. Next we compute the communication cost of an $(n, k, \mathrm{x}, \mathcal{P})$ PARE-code.

Lemma 8. *For an $(n, k, \mathrm{x}, \mathcal{P})$ PARE-code, let $||\bar{P}_j||_0 = \sum_{i:N_i \in \bar{P}_j} \mathbb{1}_{\{x_i \neq 0\}}$. For each $\mathcal{P}$-patterned erasure $N_{\mathcal{I}}$, define $j^*_{\mathcal{I}} = \underset{j}{\operatorname{argmin}}\{||\bar{P}_j||_0 \,\big|\, N_{\mathcal{I}} \subseteq P_j\}$ and $P(P_j) = P(j^*_{\mathcal{I}} = j)$. Then the communication cost of an $(n, k, \mathrm{x}, \mathcal{P})$ PARE-code is given by*

$$\Lambda_{\mathcal{C}} = \sum_{j=1}^{|\mathcal{P}|} ||\bar{P}_j||_0 P(P_j). \tag{4.1}$$

Proof. Let $N_{\mathcal{I}}$ be a $\mathcal{P}$-*patterned* erasure such that $N_{\mathcal{I}} \subseteq P_j$. By the nature of construction of the $(n, k, \mathrm{x}, \mathcal{P})$ PARE-code, $N_{\mathcal{I}}$ can be corrected by accessing all active nodes in $\bar{P}_j$. Since $N_{\mathcal{I}}$ can be a subset of multiple P_j, to have the minimum communication cost, we access $P_{j^*_{\mathcal{I}}}$. This accounts for a communication cost $\Lambda_S(N_{\mathcal{I}}) = ||\bar{P}_j||_0$, where $j = j^*_{\mathcal{I}}$. Thus we can write the average communication cost

$$\Lambda_{\mathcal{C}} = E_{N_{\mathcal{I}}}[\Lambda_S(N_{\mathcal{I}})] = \sum_{j=1}^{|\mathcal{P}|} E_{N_{\mathcal{I}}}[\Lambda_S(N_{\mathcal{I}}) \big| j^*_{\mathcal{I}} = j] P(j^*_{\mathcal{I}} = j) = \sum_{j=1}^{|\mathcal{P}|} ||\bar{P}_j||_0 P(P_j).$$

□

When a lower limit is placed on the storage requirement at each node, the communication cost of blockchain systems using $(n, \mathrm{x}^*, k^*, \mathcal{P})$ PARE-codes grows larger than that of coded sharding. The corresponding result is precisely stated in the following Lemma.

Lemma 9. *For blockchain systems with n nodes that uses an $(n, \mathrm{x}, k, \mathcal{P})$ PARE-code with non zero lower limit on storage, i.e, $x_i \geq l \cdot k > 0, 1 \leq i \leq n$, the communication cost $\Lambda_{\mathcal{C}} \geq n - t_{max}(\mathcal{P})$.*

Proof. When there is a non-zero lower limit on the storage at each node, $||\bar{P}_j||_0 = |\bar{P}_j|$. Thus

$$\Lambda_S = \sum_{j=1}^{|\mathcal{P}|} |\bar{P}_j| P(P_j) \geq (\min_j |\bar{P}_j|) \sum_{j=1}^{|\mathcal{P}|} P(P_j) = n - t_{max}(\mathcal{P}).$$

□

Thus the above lemma shows that, when a lower limit is placed on the storage at each node, the communication cost in PARE-codes is always greater than that of coded sharding. Also, the communication cost given by $\Lambda_{\mathcal{C}} = \sum_{j=1}^{|\mathcal{P}|} |\bar{P}_j| P(P_j)$ only depends on the system properties and does not depend on any design parameters (x, k). Thus for a fixed n it is not possible to improve trade off between storage efficiency and communication cost in PARE-codes when there is a lower limit on storage. Next we design codes which are *$\mathcal{P}$-correcting* and have the additional property that it can correct single node failures locally. These codes have an additional locality parameter r that allows to improve the trade off between storage efficiency and communication cost.

4.2 PARE-LRC: Locally recoverable PARE-codes

First we define the notion of local recovery in the context of PARE-codes. Since different nodes store different number of coded shards, when a single node fails, it results in multiple shards failing. This scenario is thus different from the conventional local recovery schemes that correct all single node failures locally. We extend the conventional notion of local

recovery in the context of PARE-codes by going back to viewing an $(n, k, \mathrm{x}, \mathcal{P})$ PARE-code as an $(\sum_{i=1}^{n} x_i, k)$ linear code. So when a single node fails, it implies that either the first x_1 symbols are in erasures, or the second x_2 symbols are in erasures, or the third x_3 are in erasures, and so on. We are interested in $\mathcal{P}$-*correcting* codes that correct these erasures by contacting at most r other information symbols or shards. We call such codes $(n, k, r, \mathrm{x}, \mathcal{P})$ PARE-LRC codes. More formally, we have the following definition for an $(n, k, r, \mathrm{x}, \mathcal{P})$ PARE-LRC code:

Definition 4. *Let a blockchain system have n nodes and patterned set $\mathcal{P}$. A code is said to be $(n, k, r, \mathrm{x}, \mathcal{P})$ PARE-LRC if it satisfies the following properties:*
1) It is an $(n, k, \mathrm{x}, \mathcal{P})$ PARE-code.
2) For every $x_i \neq 0$ shard positions (as described earlier) there exists a set of shards R_{x_i} not containing the x_i shards and of size at most r shards (i.e., $|R_{x_i}| \leq r$), such that the x_i shards can be repaired by just the knowledge of the R_{x_i} shards.

Parameter R_{x_i} is called the local recovery group for the x_i shards.

An $(n, k, r, \mathrm{x}, \mathcal{P})$ PARE-LRC code can correct all single node failures by utilizing the information from at most r other shards. The following example illustrates the above definition.

Example 4. *Following is an example showing an $(n = 6, k = 3, r = 2, \mathrm{x}, \mathcal{P})$ PARE-LRC code with $\mathrm{x} = (2, 1, 0, 2, 1, 3)$ and $\mathcal{P} = \{\{N_1, N_3, N_4, N_5\}, \{N_1, N_3, N_6\}, \{N_2, N_3, N_5, N_6\}, \{N_1, N_2, N_4\}, \{N_4, N_6\}\}$ for a blockchain system with 6 nodes $\{N_1, N_2, \ldots, N_6\}$:*

N_1		N_2	N_3	N_4		N_5	N_6		
a	b	a+b	−	b	c	b+c	a	c	a+c
c_1	c_2	c_3	−	c_4	c_5	c_6	c_7	c_8	c_9

Table 4.1: Coded Shards stored at $\{N_1, N_2, \ldots, N_6\}$ using PARE-LRC code in Example 4.

In the above code the blockchain is partitioned into 3 shards $\{a, b, c\}$. Clearly, the code is

$\mathcal{P}$-correcting and has the following local recovery groups: $R_{x_1} = \{c_3, c_4\}$, $R_{x_2} = \{c_1, c_2\}$, $R_{x_4} = \{c_2, c_6\}$, $R_{x_5} = \{c_2, c_5\}$, $R_{x_6} = \{c_1, c_5\}$.

The next lemma upper bounds the communication cost for an $(n, k, r, \mathrm{x}, \mathcal{P})$ PARE-LRC code.

Lemma 10. *For an $(n, k, r, \mathrm{x}, \mathcal{P})$ PARE-LRC code, let $||\bar{P}_j||_0$, $j^*_{\mathcal{I}}$ and $P(P_j)$ be defined according to Lemma 8. Also, define $P(S, P_j) = P(|N_{\mathcal{I}}| = 1, j^*_{\mathcal{I}} = j)$ and $r_j^{\min} = \min(r, ||\bar{P}_j||_0)$. The communication cost for the code satisfies the following:*

$$\Lambda_S \leq \sum_{j=1}^{|\mathcal{P}|} [r_j^{\min} P(S, P_j) + (P(P_j) - P(S, P_j))||\bar{P}_j||_0] \tag{4.2}$$

Proof. We write:

$$\begin{aligned}
\Lambda_S = E_{N_{\mathcal{I}}}[\Lambda_S(N_{\mathcal{I}})] &= \sum_{j=1}^{|\mathcal{P}|} E_{N_{\mathcal{I}}}[\Lambda_S(N_{\mathcal{I}})\big| j^*_{\mathcal{I}} = j] P(j^*_{\mathcal{I}} = j) \\
&= \sum_{j=1}^{|\mathcal{P}|} E_{N_{\mathcal{I}}}[\Lambda_S(N_{\mathcal{I}})\big| j^*_{\mathcal{I}} = j, |N_{\mathcal{I}}| = 1] P(P_j, |N_{\mathcal{I}}| = 1) \\
&+ \sum_{j=1}^{|\mathcal{P}|} E_{N_{\mathcal{I}}}[\Lambda_S(N_{\mathcal{I}})\big| j^*_{\mathcal{I}} = j, |N_{\mathcal{I}}| > 1] P(P_j, |N_{\mathcal{I}}| > 1) \\
&\leq \sum_{j=1}^{|\mathcal{P}|} [r_j^{\min} P(S, P_j) + (P(P_j) - P(S, P_j))||\bar{P}_j||_0],
\end{aligned} \tag{4.3}$$

where the inequality arises from the fact that all single node failures in an $(n, k, r, \mathrm{x}, \mathcal{P})$ PARE-LRC code can be recovered by at most r other shards, which requires contacting at most r nodes. For $j^*_{\mathcal{I}} = j$, the failed nodes can also be recovered by contacting active nodes in $\bar{P}_j$ which there are $||\bar{P}_j||_0$ of. Thus $E_{N_{\mathcal{I}}}[\Lambda_S(N_{\mathcal{I}})\big| j^*_{\mathcal{I}} = j, |N_{\mathcal{I}}| = 1] \leq \min(r, ||\bar{P}_j||_0) = r_j^{\min}$. □

We call $P(S, P_j)$ the probability of single node failure of pattern type P_j. In situations when $P(S, P_j)$ higher than $(P(P_j) - P(S, P_j))$ (which is the probability of multiple node failure of pattern type P_j), the communication cost of PARE-LRC codes is expected to be lower than PARE-codes if r is chosen such that $r \leq ||\bar{P}_j||_0$ for all j.

Remark 10. *For the case when there is a lower limit on storage, we would like* $r \leq |\bar{P}_j|$ *for all* j *i.e.,* $r \leq n - t_{max}(\mathcal{P})$.

In rest of the thesis, we take the communication cost of a PARE-LRC code as the upper bound provided by (4.3) and try to reduce this bound. Next we provide design methods for PARE-LRC codes and method of finding good parameters (k, r, x) which provide a better communication cost compared to PARE-codes and coded sharding.

4.3 Construction of PARE-LRC codes

The next Theorem provides a sufficient condition on parameters (k, r, x) for them to form a valid PARE-LRC code.

Theorem 2. *Consider a blockchain system with* n *nodes* $\{N_1, N_2, \ldots, N_n\}$ *and patterned set* $\mathcal{P} = \{P_1, P_2, \ldots, P_{|\mathcal{P}|}\}$. *For the parameters* $(k, \tilde{r}, \mathrm{x})$, *let*

$$C(\tilde{r}, \mathrm{x}) = (\sum_{i=1}^{n} x_i) \bmod (\tilde{r} + \max(\mathrm{x})).$$

Also, let $L(k, \tilde{r}, \mathrm{x})$ *be defined as*

$$L(k, \tilde{r}, \mathrm{x}) = \begin{cases} 1 \quad \textit{if} \quad C(\tilde{r}, \mathrm{x}) = 0 \quad \textit{OR} \\ \qquad C(\tilde{r}, \mathrm{x}) - \max(\mathrm{x}) \geq k \bmod (\tilde{r}) > 0, \\ 0 \quad \textit{otherwise.} \end{cases}$$

Now if the parameters $(k, \tilde{r}, \mathrm{x})$ *(with* $x_i \leq k \ \forall i$*) satisfy conditions* (4.4) *and* (4.5) *then for the parameters* $(k, \tilde{r}, \mathrm{x})$ *there exists an* $(n, k, r = \tilde{r}\max(\mathrm{x}), \mathrm{x}, \mathcal{P})$ *PARE-LRC code,*

$$\sum_{i:N_i \in \bar{P}_j} x_i \geq k + \left(\left\lceil \frac{k}{\tilde{r}} \right\rceil - 1 \right) \max(\mathrm{x}) \ \ \forall P_j, \tag{4.4}$$

$$L(k, \tilde{r}, \mathrm{x}) = 1. \tag{4.5}$$

Proof. To see why the above theorem is true, we go back to our view of an $(n, k, \mathrm{x}, \mathcal{P})$ PARE-code as an $(\sum_{i=1}^{n} x_i, k)$ linear code. Condition (4.4) implies:

$$\sum_{i=1}^{n} x_i - k + 1 - \Big(\Big\lceil \frac{k}{\tilde{r}} \Big\rceil - 1 \Big) \max(\mathrm{x}) \geq \max_{j \in 1,2,\ldots,|\mathcal{P}|} \Big(\sum_{i:N_i \in P_j} x_i \Big) + 1.$$

The left hand side of the above equation is the minimum distance bound of $(\sum_{i=1}^{n} x_i, k)$ codes where all the symbols have $(r = \tilde{r},\ \delta = \max(\mathrm{x}) + 1)$ locality [PKL12], [SRK13][1]. Condition (4.5) is satisfied when either $(\sum_{i=1}^{n} x_i) \bmod (\tilde{r} + \max(\mathrm{x})) = 0$ OR $(\sum_{i=1}^{n} x_i) \bmod (\tilde{r} + \max(\mathrm{x})) - \max(\mathrm{x}) \geq k \bmod (\tilde{r}) > 0$. This ensures that for the parameter set $(k, \tilde{r}, \mathrm{x})$, there exists an explicit code construction (provided in [SRK13]) for the $(\sum_{i=1}^{n} x_i, k)$ linear code with $(r = \tilde{r},\ \delta = \max(\mathrm{x}) + 1)$ locality to achieve this minimum distance bound. Assume that the $(\sum_{i=1}^{n} x_i, k)$ code (that satisfies conditions (4.4) and (4.5)) is designed using the construction provided in [SRK13] to have a minimum distance of $\sum_{i=1}^{n} x_i - k + 1 - \Big(\lceil \frac{k}{\tilde{r}} \rceil - 1 \Big) \max(\mathrm{x})$.

Now, $\max_{j \in 1,2,\ldots,|\mathcal{P}|} \Big(\sum_{i:N_i \in P_j} x_i \Big)$ is the maximum number of erasures in the $(\sum_{i=1}^{n} x_i)$ codewords space that can result from $\mathcal{P}$-*patterned* erasures. Thus condition (4.4) essentially implies that the minimum distance of the $(\sum_{i=1}^{n} x_i, k)$ code (constructed as argued above) is one more than the maximum number of erasures. Thus the $(\sum_{i=1}^{n} x_i, k)$ code corrects all erasures (in the $\sum_{i=1}^{n} x_i$ codeword space) resulting from $\mathcal{P}$-*patterned* erasures and hence is $\mathcal{P}$-*correcting*. Now as pointed out in Remark 8 we construct an $(n, k, \mathrm{x}, \mathcal{P})$ PARE-code from the $\mathcal{P}$-*correcting* $(\sum_{i=1}^{n} x_i, k)$ code using step 2) of code construction 2.

Reference [RMV15] showed that codes with $(\tilde{r}, \delta)$ locality achieve $(r = \tilde{r}l, l = \delta - 1)$ cooperative locality. This implies that for all l codeword (of the $(\sum_{i=1}^{n} x_i, k)$ code) symbols there exists a set of r codeword symbols which can locally repair the l codeword symbols. In our case, $l = \delta - 1 = \max(\mathrm{x})$ and $r = \tilde{r} \max(\mathrm{x})$. In other words, all $\max(\mathrm{x})$ codeword symbols can be locally repaired by a set of $\tilde{r} \max(\mathrm{x})$ codeword symbols. Thus the code satisfies

[1]Codes with (r, δ) locality as defined in [PKL12] are $(r, \delta, \alpha = 1)$ LRCs according to [SRK13].

condition 2 in Definition 4 with $r = \tilde{r}\max(\mathrm{x})$ and hence is an $(n, k, r = \tilde{r}\max(\mathrm{x}), \mathrm{x}, \mathcal{P})$ PARE-LRC Code. □

It should be noted that the conditions in Theorem 2 are a set of sufficient conditions for the construction of $(n, k, r = \tilde{r}\max(\mathrm{x}), \mathrm{x}, \mathcal{P})$ PARE-LRC codes, i.e., if the parameters $(k, \tilde{r}, \mathrm{x})$ satisfy conditions (4.4) and (4.5), then we can construct an $(n, k, r = \tilde{r}\max(\mathrm{x}), \mathrm{x}, \mathcal{P})$ PARE-LRC code that can be used in a blockchain system. They are however not necessary conditions and there may exist other set of parameters that do not satisfy conditions (4.4) and (4.5) but still may be PARE-LRC codes. For instance, the code provided in Example 4 is a PARE-LRC code but it does not satisfy conditions (4.4) and (4.5).

4.4 Parameter optimization for PARE-LRC codes

In this section we provide methods to find parameters $(k, \tilde{r}, \mathrm{x})$ which give $(n, k, r = \tilde{r}\max(\mathrm{x}), \mathrm{x}, \mathcal{P})$ PARE-LRC codes with best possible trade off between $\Lambda_{\mathcal{C}}$ and $\Gamma_{\mathcal{C}}$. Since conditions (4.4) and (4.5) do not define convex sets in $(k, \tilde{r}, \mathrm{x})$, it is not possible to formulate an integer optimization problem in variables $(k, \tilde{r}, \mathrm{x})$ to find optimal set of parameters (like we did for PARE-codes). However for fixed k and $\tilde{r}$, condition (4.4) is convex in x. For fixed k and $\tilde{r}$, we solve the following problem which we call $ILP(k, \tilde{r})$:

$$\begin{aligned} \min_{\mathrm{x}} \; Obj(\mathrm{x}; k, \tilde{r}) = \; & \frac{B}{n}\frac{1^T\mathrm{x}}{k} + \lambda\tilde{r}\max(\mathrm{x}) \\ s.t. \sum_{i:N_i\in\bar{P}_j} x_i \geq k + \Big(\Big\lceil\frac{k}{\tilde{r}}\Big\rceil - 1\Big)\max(\mathrm{x}), j = 1, 2, \ldots, |\mathcal{P}| \\ l\cdot k \leq x_i \leq k\cdot u, i = 1, 2, \ldots, n, x_i \in \mathrm{Z}^+, i = 1, 2, \ldots, n, \end{aligned} \tag{4.6}$$

where $\lambda > 0$ is a parameter that controls the importance between communication cost and storage efficiency. Let the optimal solution and optimal value of $ILP(k, \tilde{r})$ be $\mathrm{x}^{ILP}(k, \tilde{r})$ and $T^{ILP}(k, \tilde{r})$ $\big(= Obj(\mathrm{x}^{ILP}(k, \tilde{r}); k, \tilde{r})\big)$ respectively. If the problem is infeasible we assume $T^{ILP}(k, \tilde{r})$ is infinity.

Now, if $ILP(k, \tilde{r})$ is feasible and the solution $\mathrm{x}^{ILP}(k, \tilde{r})$ does not satisfy condition (4.5), we sequentially increase the value of some coordinates of $\mathrm{x}^{ILP}(k, \tilde{r})$ $\big($to obtain $\mathrm{x}^{mod}(k, \tilde{r})\big)$ starting from the coordinates having the lowest value without changing $\max(\mathrm{x}^{ILP}(k, \tilde{r}))$ until it satisfies condition (4.5). We call this procedure $\Pi(\mathrm{x}; k, \tilde{r})$ which operates on $\mathrm{x} = \mathrm{x}^{ILP}(k, \tilde{r})$ and returns $\mathrm{x}^{mod}(k, \tilde{r})$ and $T^{mod}(k, \tilde{r})\big(= Obj(\mathrm{x}^{mod}(k, \tilde{r}); k, \tilde{r})\big)$. If no such increase of $\mathrm{x}^{ILP}(k, \tilde{r})$ is possible, $\Pi(\mathrm{x}; k, \tilde{r})$ sets $T^{mod}(k, \tilde{r})$ as infinity. If $\mathrm{x}^{ILP}(k, \tilde{r})$ satisfies condition (4.5), then we set $\mathrm{x}^{mod}(k, \tilde{r}) = \mathrm{x}^{ILP}(k, \tilde{r})$ and $T^{mod}(k, \tilde{r}) = T^{ILP}(k, \tilde{r})$.

The following guide our choice of k and $\tilde{r}$ for which $ILP(k, \tilde{r})$ is solved.

Remark 11. *Since $x_i^{mod} \geq 1$ (as x_i is a positive integer when a lower limit is placed on storage), $\Gamma_{\mathcal{C}} = \frac{nk}{\sum_1^n x_i} \leq k$. Thus for $k \leq n - t_{max}(\mathcal{P})$, $\Gamma_{\mathcal{C}}$ for PARE-LRC code with parameters $(k, \tilde{r}, \mathrm{x}^{mod}(k, \tilde{r}))$ is always less than that of coded sharding. Thus we always choose $k \geq n - t_{max}(\mathcal{P})$.*

Remark 12. *As we are looking to reduce communication cost when a lower limit is placed on the storage at each node, according to Remark 10, we would like $r = \tilde{r} \max(\mathrm{x}) \leq n - t_{max}(\mathcal{P})$. This is not possible if $\tilde{r} > n - t_{max}(\mathcal{P})$ since $\max(\mathrm{x}) \geq 1$. Thus we choose $\tilde{r} \leq n - t_{max}(\mathcal{P})$.*

Lemma 11. *For a given $\tilde{r}$ and upper and lower limits u and l on the storage at each node (i.e., $u \cdot k \geq x_i \geq l \cdot k, 1 \leq i \leq n$), $ILP(k, \tilde{r})$ is infeasible if $k > a(\tilde{r})$ where $a(\tilde{r}) = \tilde{r}[1 + \frac{u(n - t_{max}(\mathcal{P}) - 1)}{l}]$.*

Proof. For any x that satisfies $u \cdot k \geq x_i \geq l \cdot k, 1 \leq i \leq n$, the following holds:

$$
\begin{aligned}
& k > \tilde{r}[1 + \frac{u(n - t_{max}(\mathcal{P}) - 1)}{l}] \\
\implies & 1 + l\Big(\frac{k}{\tilde{r}} - 1\Big) > u(n - t_{max}(\mathcal{P})) \\
\implies & k + l \cdot k\Big(\left\lceil \frac{k}{\tilde{r}} \right\rceil - 1\Big) > u \cdot k(n - t_{max}(\mathcal{P})) \\
\implies & k + \max(\mathrm{x})\Big(\left\lceil \frac{k}{\tilde{r}} \right\rceil - 1\Big) > u \cdot k(\min_j |\bar{P}_j|) \\
\implies & k + \max(\mathrm{x})\Big(\left\lceil \frac{k}{\tilde{r}} \right\rceil - 1\Big) > \min_j \Big(\sum_{i: N_i \in \bar{P}_j} x_i\Big),
\end{aligned}
$$

where $\min_j$ is for $1 \leq j \leq |\mathcal{P}|$.

The above implies that there is at least one P_j for which

$$\sum_{i:N_i \in \bar{P}_j} x_i < k + \left(\left\lceil \frac{k}{\tilde{r}} \right\rceil - 1 \right) \max(\mathrm{x}),$$

and hence the first constraint in $ILP(k, \tilde{r})$ cannot be satisfied. □

Thus noting Remarks 11, 12 and Lemma 11, we solve $ILP(k, \tilde{r})$ using the procedure described earlier (for obtaining $\mathrm{x}^{mod}(k, \tilde{r})$) for all $\tilde{r}$ in $\{1, 2, \ldots, (n - t_{max}(\mathcal{P}))\}$ and k in $\{(n - t_{max}(\mathcal{P})), \ldots, a(\tilde{r})\}$ and pick the pair which gives us the best $T^{mod}(k, \tilde{r})$. The procedure is summarized in Algorithm 1.

Algorithm 1 Algorithm for parameter optimization of $(n, k, r = \tilde{r}\max(\mathrm{x}), \mathrm{x}, \mathcal{P})$ PARE-LRC codes.

$R_{range} = \{1, 2, \ldots, (n - t_{max}(\mathcal{P}))\}$
for $\tilde{r}$ in R_{range} **do**
    $K_{range} = \{(n - t_{max}(\mathcal{P})), \ldots, a(\tilde{r})\}$
    for k in K_{range} **do**
        $[\mathrm{x}^{ILP}(k, \tilde{r}),\ T^{ILP}(k, \tilde{r})] \leftarrow$ *Solve* $ILP(k, \tilde{r})$
        if $\mathrm{x}^{ILP}(k, \tilde{r})$ satisfies condition (4.5) **then**
            $\mathrm{x}^{mod}(k, \tilde{r}) \leftarrow \mathrm{x}^{ILP}(k, \tilde{r})$
            $T^{mod}(k, \tilde{r}) \leftarrow T^{ILP}(k, \tilde{r})$
        else
            $[\mathrm{x}^{mod}(k, \tilde{r}), T^{mod}(k, \tilde{r})] \leftarrow \Pi(\mathrm{x}^{ILP}(k, \tilde{r}); k, \tilde{r})$
        end if
    end for
end for
$[k^{opt}, \tilde{r}^{opt}] \leftarrow \underset{k,\tilde{r}}{\operatorname{argmin}}\, T^{mod}(k, \tilde{r})$
Output: $[k^{opt}, \tilde{r}^{opt}, \mathrm{x}^{mod}(k^{opt}, \tilde{r}^{opt})]$

Although Algorithm 1 requires solving $ILP(k, \tilde{r})$ for all $\tilde{r}$ in $\{1, 2, \ldots, (n - t_{max}(\mathcal{P}))\}$ and k in $\{(n - t_{max}(\mathcal{P})), \ldots, a(\tilde{r})\}$, simulations show us that $(k^{opt}, \tilde{r}^{opt})$ obtained by solving the $ILP(k, \tilde{r})$ for much smaller ranges of k and $\tilde{r}$ still gives an $(n, k, \tilde{r}\max(\mathrm{x}), \mathrm{x}, \mathcal{P})$ PARE-LRC code which has lower communication cost and a better trade-off between storage efficiency and communication cost compared to coded sharding and PARE-codes.

We summarize the above discussion on optimization of parameter set $(k, \tilde{r}, \mathrm{x})$ and construction of $(n, k, r = \tilde{r}\max(\mathrm{x}), \mathrm{x}, \mathcal{P})$ PARE-LRC code in the following code construction procedure.

Code Construction 3. ***(PARE-LRC)*** *For a blockchain of size B with n nodes having erasure patterned set $\mathcal{P} = \{P_1, P_2, \ldots, P_{|\mathcal{P}|}\}$, let the blockchain be partitioned into k shards $s_1, s_2, \ldots, s_k$. Let the upper and lower limits on the storage at each node be u and l respectively. Our code construction for improved storage efficiency and communication cost trade off involves the following steps:*

1. *Solve Algorithm 1 to get $k^{opt}, \tilde{r}^{opt}$ and x^{mod}.*

2. *Design an $(\sum_{i=1}^{n} x_i^{mod}, k^{opt})$ code that achieves the minimum distance bound*

$$d_{min} = \sum_{i=1}^{n} x_i^{mod} - k^{opt} + 1 - \left(\left\lceil \frac{k^{opt}}{\tilde{r}^{opt}} \right\rceil - 1 \right) \max(\mathrm{x}^{mod})$$

   where all the symbols have ($r = \tilde{r}^{opt}$, $\delta = \max(\mathrm{x}^{mod}) + 1$) locality using the construction provided in [SRK13].

3. *Partition the blockchain into k^{opt} shards $s_1, s_2, \ldots, s_{k^{opt}}$. Generate $(\sum_{i=1}^{n} x_i^{mod})$ coded shards from these k^{opt} shards using the above $(\sum_{i=1}^{n} x_i^{mod}, k^{opt})$ code. Sequentially allocate these coded shards to all the nodes with node N_i getting x_i^{mod} coded shards.*

4.5 Discussion

In Chapter 4, we extent the notion of local recovery to the context of PARE-codes to result in PARE-LRC codes. In Theorem 2 we provide a set of sufficient conditions for the design of PARE-LRC codes. Algorithm 1 provides a way to optimize the parameters of these codes to result in the best possible trade off between storage efficiency and communication cost. In Chapter 6, we simulate the PARE-LRC codes designed in this chapter to see their performance benefits in terms of trade off between storage efficiency and communication cost. In the next chapter we will look at the redesign complexity of PARE-codes.

CHAPTER 5

Redesign Complexity Analysis

As pointed out in Chapter 1, a drawback of coded sharding is that the (n, k) code used depends on the number of nodes currently in the system. If a new node is added or it leaves the system, a new (n', k') code (where n, n' and k, k' are not necessarily the same) has to be designed and the coded shards have to be regenerated. In this chapter we analyze the redesign complexity of blockchain systems that use the codes constructed using construction 1 and 2. Codes constructed using construction 1 and 2 depend on the parameter (x^*, k^*) that are the optimal solution of problem (3.1). As nodes leave or are added to the blockchain system, the erasure patterned set changes and thus the value of these parameters may possibly change and hence the coded shards stored at different nodes might have to redesigned. Next we look at the situation of adding or removing a single node from the blockchain system and analyze cases where the old choice of parameters is still optimal after adding or removing the new node. We observe that when the blockchain follows the model of periodic node failures as described in Section 2.2 with no upper or lower limit on storage, and has a sufficiently large number of nodes, with high probability the old choice of parameters will still be optimal after adding or removing the new node. Thus no redesign will be necessary in the system with high probability when a new node gets added or is removed from the system. In situations when the old choice of parameters is not optimal, redesign is necessary but his happens with a very low probability.

5.1 Effect of adding a new node

Consider a blockchain system with n nodes $\{N_1, N_2, \ldots, N_n\}$. Let $\mathcal{P}_{old} = \{P_1, P_2, \ldots, P_{|\mathcal{P}_{old}|}\}$ be the patterned erasure set for these n nodes. When a new node (N_{n+1}) joins the system, the patterned erasure set changes to $\mathcal{P}_{new}$. For each pattern P_j in $\mathcal{P}_{old}$, the patterns in this new set $\mathcal{P}_{new}$ must either be of the form (P_j, N_{n+1}) where the new node also fails with the existing nodes in P_j, or simply P_j, when the new node does not fail with the existing nodes in P_j. Another possibility is when all the n nodes $\{N_1, N_2, \ldots, N_n\}$ are active, and N_{n+1} node fails, thus resulting in an additional erasure pattern. Hence, $\mathcal{P}_{new}$ is a subset of $\mathcal{P}^g_{new}$ $= \{P_1, \ldots, P_{|\mathcal{P}_{old}|}, (P_1, N_{n+1}), (P_2, N_{n+1}), \ldots, (P_{|\mathcal{P}_{old}|}, N_{n+1}), N_{n+1}\}$ with the property that for each pattern P_j either P_j or (P_j, N_{n+1}) is present in $\mathcal{P}_{new}$.

When nodes follow the model of periodic node failures as described in Section 2.2 and with no upper or lower limit on storage, we will show that with high probability no redesign is needed for a blockchain system that uses codes constructed using construction 1 and 2.

Lemma 12. *For a blockchain with n nodes, erasure patterned set $\mathcal{P}_{old}$, and no lower or upper limit on storage, let $(\mathrm{x}^{old}, k^{old})$ be the optimal solution to problem (3.1). If the (u, d, ϕ) pattern of the new node (N_{n+1}) joining the system is same as any of the previous n nodes, $((\mathrm{x}^{old}, 0), k^{old})$ is an optimal solution to the new $(n+1)$ node system with patterned set $\mathcal{P}_{new}$.*

Proof. Assume that N_l and N_{n+1} have the same (u, d, ϕ) pattern. Thus we have $\mathcal{P}_{new} = \{\underset{\sim}{P}_1, \underset{\sim}{P}_2, \ldots, \underset{\sim}{P}_{|\mathcal{P}_{old}|}\}$ where $\underset{\sim}{P}_i = P_i$ if $N_l \notin P_i$ and $\underset{\sim}{P}_i = P_i \cup N_{n+1}$ if $N_l \in P_i$. This means that in the new patterned set $\mathcal{P}_{new}$, N_{n+1} and N_l either appear together in the sets $\underset{\sim}{P}_j$ or they do not appear at all. In this case, the modified problem (3.1) which we solve to get the optimal solution after the addition of new node becomes :

$$\min_{x_1,\ldots,x_{n+1}} \frac{B}{n+1}\frac{\sum_{i=1,i\neq l}^{n} x_i + (x_l + x_{n+1})}{k}$$
$$s.t. \sum_{i:N_i\in\bar{P}_j, i\neq l, n+1} x_i + \mathbb{1}_{N_l\in\bar{P}_j}(x_l + x_{n+1}) \geq k, \;\; j = 1, 2, \ldots, |\mathcal{P}_{old}| \tag{5.1}$$
$$x_i \in \mathrm{Z}^+, i = 1, 2, \ldots, n$$
$$k \in \mathrm{Z}^{++}.$$

Note that with regards to Remark 2, we have removed the constraint $x_i \leq k$ from the problem. Now calling $(x_l + x_{n+1})$ a single variable $\bar{x}_l$, the above problem becomes exactly the same as problem (3.1) (with the constraint $x_i \leq k$ removed, and a different scaling constant in the objective) in variables $(x_1, x_2, \ldots, \bar{x}_l, \ldots, x_n)$, which yields the solution $(\mathrm{x}^{old}, k^{old})$. Thus $x_i = x_i^{old}, i \neq l$, $\bar{x}_l = x_l^{old}$ and $k = k^{old}$. Now, $x_{n+1} = 0$, $x_l = x_l^{old}$, is feasible and attains the same optimal objective and hence is an optimal solution. □

Remark 13. *It should be noted that for any optimal solution* $(\mathrm{x}^* = (x_1^*, x_2^*, \ldots, x_{n+1}^*), k^*)$ *of problem* (5.1)*,* $x_l^* + x_{n+1}^* \leq k^*$ *must hold. If the sum was greater than* k^**, we can reduce the sum* $x_l^* + x_n^*$ *to be equal to* k^* *keeping* k^* *constant. After this reduction, the new point still remains feasible and the optimization objective value decreases. This is a contradiction to the fact that* (x^*, k^*) *is an optimal solution. Thus we can call* $x_l + x_{n+1}$ *as the new variable* $\bar{x}_l$ *which is implicitly constrained to be always less than* k^* *at the optimal solution point.*

Remark 14. *In the above lemma, since* $((\mathrm{x}^{old}, 0), k^{old})$ *is the optimal solution to problem* (3.1) *for the* $(n+1)$ *blockchain system, we achieve the following:*

1. *The number of shards that blockchain must be partitioned after a new node joins the system is the same as the original number of shards the blockchain was already partitioned into before the* $(n+1)^{st}$ *node was added.*

2. *The new node need not store any coded shards (since* $x_{n+1} = 0$*) and the number of coded shards that needs to be stored at the first* n *nodes is the same as before.*

3. *Since the original set of coded shards stored at the first n nodes designed using code construction 1 and 2 was able to correct all $\mathcal{P}_{old}$-patterned erasures, and the new node is not storing any coded shards, these same coded shards will still be able to correct all the $\mathcal{P}_{new}$-patterned erasures that arise after the addition of the new node.*

Thus the above lemma indicates that if the newly added node has the same periodicity pattern as one of the previous nodes already present in the system, it need not store any data; existing nodes can store the same coded shards as they initially stored, and the blockchain will still have maximum storage efficiency. Maximum storage efficiency is due to the fact that we are using an optimal solution $((\mathrm{x}^{old}, 0), k^{old})$ to problem (3.1) for the $(n+1)$ blockchain system with patterned set $\mathcal{P}_{new}$. Next we show that the probability of this happening tends to one as the number of nodes increases in the system.

Theorem 3. *Consider a blockchain system with n nodes, erasure patterned set $\mathcal{P}_{old}$, and no lower or upper limit on storage. Suppose the system follows the model of periodic node failures described in Section 2.2 with a fixed set of uptime-downtime pairs (U, D). Also let $(\mathrm{x}^{old}, k^{old})$ be an optimal solution to problem (3.1) for the system having n nodes. After the arrival of the $(n+1)^{st}$ node, the probability that $((\mathrm{x}^{old}, 0), k^{old})$ is an optimal solution for problem (3.1) with $(n+1)$ nodes and patterned set $\mathcal{P}_{new}$ tends to one as $n \to \infty$.*

Proof. Let n_i be the number of nodes initially in the blockchain system who have the periodicity pattern (u_i, d_i), $1 \leq i \leq e$. Since each node picks an uptime-downtime pair uniformly at random from (U, D), as the number of nodes increases in the system, $n_i \approx \frac{n}{e}$. Call the event that $((\mathrm{x}^{old}, 0), k^{old})$ is optimal after the addition of the $(n+1)^{st}$ node as Y, and the event that the $(n+1)^{st}$ node has same (u, d, ϕ) pattern as any of previous n nodes in the blockchain system as A. Also, call the event that the $(n+1)^{st}$ node has uptime-downtime pair as (u_i, d_i) from (U, D) as S_i. From Lemma 12, we have:

$$\mathrm{Prob(Y)} \geq \mathrm{Prob(A)} = 1 - \mathrm{Prob}(\bar{\mathrm{A}}) = 1 - \sum_{i=1}^{e} \frac{1}{e} \mathrm{Prob}(\bar{\mathrm{A}}|\mathrm{S_i}).$$

Here, $\bar{\text{A}}$ is the event that the $(n+1)^{th}$ node has a different (u, d, ϕ) pattern than any of the previous n nodes. Now conditioned on the event S_i, the $(n+1)^{th}$ node has different (u, d, ϕ) pattern if it has a different pattern than the n_i nodes who have a pattern (u_i, d_i). Since for each (u_i, d_i) pattern there is a random phase $\phi_i \in [0, u_i]$, we can write $\text{Prob}(\bar{\text{A}}|\text{S}_\text{i}) = (\frac{u_i}{u_i+1})^{n_i}$. Thus $\text{Prob(Y)} \geq 1 - \sum_{i=1}^{e} \frac{1}{e}\big(\frac{u_i}{u_i+1}\big)^{n_i}$. Now as n increases, $n_i \approx \frac{n}{e}$ and thus $(\frac{u_i}{u_i+1})^{\frac{n}{e}}$ tends to zero which means that Prob(Y) tends to one. □

Thus for a blockchain system following the model of periodic node failures described in Section 2.2, with regards to Remark 14, with high probability no redesign of the system is needed when the system is storing coded shards designed using code construction 1 and 2.

5.2 Effect of a node leaving

Here we consider a blockchain system with n nodes having an erasure patterned set $\mathcal{P}_{old} = \{P_1, P_2, \ldots, P_{|\mathcal{P}_{old}|}\}$. We assume that the n^{th} node (N_n) permanently leaves the system. The resulting patterned erasure set for the $(n-1)$ system becomes $\mathcal{P}_{new} = \{\underset{\sim}{P}_1, \underset{\sim}{P}_2, \ldots, \underset{\sim}{P}_{|\mathcal{P}_{old}|}\}$ where $\underset{\sim}{P}_j = P_j$ if $N_n \notin P_j$ and $\underset{\sim}{P}_j = P_j \setminus N_n$ if $N_n \in P_j$. We assume the non-trivial case of N_n storing non-zero number of coded shards so that when it leaves, the resulting system may cease to be $\mathcal{P}_{new}$-correcting.

With this setting we will show that, when nodes follow the model of periodic node failures as described in Section 2.2 and with no lower or upper limit on storage, for codes constructed using construction 1 and 2 no redesign of the existing coded shards stored at the first $(n-1)$ nodes will be needed. However, to make the system $\mathcal{P}_{new}$-correcting some additional coded shards have to be stored at some of the existing nodes. The following lemma is analogous to Lemma 12 for this case.

Lemma 13. *For a blockchain with n nodes, patterned erasure set $\mathcal{P}_{old}$ and no lower or upper limit on storage, let $(\text{x}^{old} = (x_1^{old}, x_2^{old}, \ldots, x_n^{old}), k^{old})$ be the optimal solution to problem*

(3.1). *If the* (u, d, ϕ) *pattern of the node* N_n *is same as node* N_l, $1 \leq l \leq n-1$, *then* $(\bar{\mathrm{x}} = (x_1^{old}, \ldots, x_l^{old} + x_n^{old}, \ldots, x_{n-1}^{old}), k^{old})$ *is an optimal solution to problem* (3.1) *for the* $(n-1)$ *system with patterned set* $\mathcal{P}_{new}$.

Proof. Since N_l and N_n have the same (u, d, ϕ) pattern, in the patterned set $\mathcal{P}_{old}$, N_n and N_l appear together in the sets P_j or they do not appear at all. Thus we can write $\underset{\sim}{P}_j = P_j$ if $N_l \notin P_j$ and $\underset{\sim}{P}_j = P_j \setminus N_l$ if $N_n \in P_j$. In this case problem (3.1) which we solve to get the optimal solution for the original system with n node becomes

$$\begin{aligned} \min_{x_1,\ldots,x_n} \ & \frac{B}{n} \frac{\sum_{i=1, i\neq l,n}^{n} x_i + (x_l + x_n)}{k} \\ s.t. \ & \sum_{i:N_i \in \bar{P}_j, i\neq l,n} x_i + \mathbb{1}_{N_l \in \bar{P}_j}(x_l + x_n) \geq k, \\ & \qquad j = 1, 2, \ldots, |\mathcal{P}_{old}| \\ & x_i \in \mathrm{Z}^+, i = 1, 2, \ldots, n \\ & k \in \mathrm{Z}^{++}. \end{aligned} \tag{5.2}$$

The above problem has an optimal solution $(\mathrm{x}^{old} = (x_1^{old}, x_2^{old}, \ldots, x_n^{old}), k^{old})$. Using a logic similar to Remark 13, we must have $x_l^{old} + x_n^{old} \leq k$. Thus $(x_l^{old} + x_n^{old})$ can possibly represent the number of coded shards stored at node N_l.

Since $\underset{\sim}{\bar{P}}_j = \{N_1, N_2, \ldots, N_{n-1}\} \setminus \underset{\sim}{P}_j$, the set $\{i : N_i \in \bar{P}_j, i \neq l, n\}$ is the same as $\{i : N_i \in \underset{\sim}{\bar{P}}_j, i \neq l\}$. Also $\mathbb{1}_{N_l \in \bar{P}_j} = \mathbb{1}_{N_l \in \underset{\sim}{\bar{P}}_j}$. Thus

$$\sum_{i:N_i \in \bar{P}_j, i\neq l,n} x_i + \mathbb{1}_{N_l \in \bar{P}_j}(x_l + x_n) = \sum_{i:N_i \in \underset{\sim}{\bar{P}}_j, i\neq l} x_i + \mathbb{1}_{N_l \in \underset{\sim}{\bar{P}}_j}(x_l + x_n).$$

Now calling $(\bar{x}_1 = x_1, \bar{x}_2 = x_2, \ldots, \bar{x}_l = x_l + x_n, \ldots, \bar{x}_{n-1} = x_{n-1})$ problem (5.2) can be modified into (with variables $(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_l, \ldots, \bar{x}_{n-1})$)

$$
\begin{aligned}
&\min_{\bar{x}_1,\ldots,\bar{x}_{n-1}} \frac{B}{n-1}\frac{\sum_{i=1}^{n-1}\bar{x}_i}{k} \\
s.t. &\sum_{i:N_i\in\bar{\underset{\sim}{P}}_j, i\neq l} \bar{x}_i + \mathbb{1}_{N_l\in\bar{\underset{\sim}{P}}_j}(\bar{x}_l) \geq k, \\
&\qquad\qquad j = 1,2,\ldots,|\mathcal{P}_{old}| \\
&\bar{x}_i \in \mathrm{Z}^+, i = 1,2,\ldots,n \\
&k \in \mathrm{Z}^{++}.
\end{aligned}
\tag{5.3}
$$

Since problem (5.3) is obtained from problem (5.2), it has an optimal solution ($\bar{x}_1 = x_1^{old}, \bar{x}_2 = x_2^{old}, \ldots, \bar{x}_l = x_l^{old} + x_n^{old}, \ldots, \bar{x}_{n-1} = x_{n-1}^{old}, k = k^{old}$), where $x_l^{old} + x_n^{old} \leq k$ as argued before. Now problem (5.3) is exactly the same as problem (3.1) for the $(n-1)$-sized blockchain with the erasure patterned set $\mathcal{P}_{new}$. Thus $(\bar{\mathrm{x}} = (x_1^{old}, \ldots, x_l^{old} + x_n^{old}, \ldots, x_{n-1}^{old}), k^{old})$ is an optimal solution to problem (3.1) for the $(n-1)$ system with patterned set $\mathcal{P}_{new}$. □

Remark 15. *In the above lemma, since* $(\bar{\mathrm{x}} = (x_1^{old}, \ldots, x_l^{old} + x_n^{old}, \ldots, x_{n-1}^{old}), k^{old})$ *is the optimal solution to problem* (3.1) *for the* $(n-1)$ *blockchain system obtained after the node* N_n *leaves the system, we achieve the following:*

1. *The number of shards that blockchain must be partitioned into after the node* N_n *leaves the system is same as the original number of shards the blockchain was already partitioned into before the node left.*

2. *All the nodes store the same number of coded shards they stored before* N_n *left except for* N_l *which stores* $x_l^{old} + x_n^{old}$ *number of coded shards i.e., the number of coded shards it stored before, along with the number of coded shards* N_n *was storing before it left.*

3. *Since the original set of coded shards stored at the first n nodes designed using code construction 1 and 2 were able to correct all* $\mathcal{P}_{old}$*-patterned erasures, storing the exact same coded shards stored at node* N_n *at* N_l *will allow these coded shards to correct all the* $\mathcal{P}_{new}$*-patterned erasures that arise after the node* N_n *has left the system.*

Regarding the last statement, consider steps 2) and 3) of code construction 2. For the solution of problem (3.1) ($\bar{\mathrm{x}} = (x_1^{old}, \ldots, x_l^{old} + x_n^{old}, \ldots, x_{n-1}^{old}), k^{old}$), since the sum $\sum_{i=1}^{n-1} \bar{x}_i = \sum_{i=1}^{n} x_i^{old}$, the same ($\sum_{i=1}^{n} x_i^{old}$,$k^{old}$) MDS code designed for the system with N_n can be used for the new system with $(n-1)$ nodes to generate $\sum_{i=1}^{n} x_i^{old}$ coded shards. Now before performing step 3) of code construction 2 consider a permutation of these generated coded shards such that the last x_n^{old} coded shards are moved after the first $\sum_{i=1}^{l-1} x_i^{old}$ coded shards. Now following step 3) procedure in code construction 2 of allocating these coded shards, the allocation will essentially be the same as just re-storing the exact same coded shards stored earlier at node N_n at N_l.

Thus from Lemma 13 and Remark 15 we see that if the node (N_n) which leaves the system has the same periodicity pattern as one of the remaining nodes in the system (say N_l, for $n \neq l$), no redesign is necessary in the sense that all the nodes apart from N_l can store the same coded shards they were storing before N_n left and N_l has to additionally store the coded shards stored by N_n along with the coded shards it stored from before. Next we show that the probability of this happening tends to one if initially a large number of nodes were present in the system. We also remark that in this case, the system will still have maximum storage efficiency, because we are using an optimal solution ($\bar{\mathrm{x}} = (x_1^{old}, \ldots, x_l^{old} + x_n^{old}, \ldots, x_{n-1}^{old}), k^{old}$) to problem (3.1) for the $(n-1)$ blockchain system with patterned set $\mathcal{P}_{new}$.

Theorem 4. *Consider a blockchain with n nodes, erasure patterned set $\mathcal{P}_{old}$, and no lower or upper limit on storage. Let it follow the model of periodic node failures described in Section 2.2 with a fixed set of uptime-downtime pairs (U, D). Also let $(\mathrm{x}^{old} = (x_1^{old}, x_2^{old}, \ldots, x_n^{old}), k^{old})$ be an optimal solution to problem (3.1) for the system having n nodes. After node N_n leaves the system, the probability that $(\bar{\mathrm{x}} = (x_1^{old}, \ldots, x_l^{old} + x_n^{old}, \ldots, x_{n-1}^{old}), k^{old})$ is an optimal solution for problem (3.1) with $(n-1)$ nodes and patterned set $\mathcal{P}_{new}$ tends to one as $n \to \infty$.*

Proof. Let A be the event that the n^{th} node has same (u, d, ϕ) pattern as any of previous $(n-1)$ nodes in the blockchain system. From the proof of Theorem 3 we know that $P(A) \to 1$

as $n \to \infty$. This result combined with Lemma 13 completes the proof. □

Thus from the above discussion, we see that for a blockchain system that follows the model of periodic node failures with no limits on storage, with high probability no redesign is necessary for codes designed using construction 1 and 2 when a node leaves or joins the blockchain system. Unfortunately, similar analysis does not hold for a blockchain system that is designed using code construction 3. This is due to the non-linear nature of conditions (4.4) and (4.5) in parameter set $(k, \tilde{r}, \mathrm{x})$ used in the design of PARE-LRC codes preventing a similar change of variables as in Lemmas 12 and 13 for this situation.

5.3 Discussion

In Chapter 5 we show that for codes constructed using construction 1 and 2, no redesign is necessary with high probability when nodes leave of are added to the system provided the number of nodes initially in the system are high. Unfortunately the analysis carried out for code constructions 1 and 2 do not naturally extend to code 3 and they have high redesign complexity. In the next chapter we look at the simulation results for PARE-codes and PARE-LRC codes.

CHAPTER 6

Simulation Results

We simulate a blockchain system with the model described in Section 2.2 for different choices of U and D. First, we look at the advantage of PARE-codes in improving the storage efficiency in blockchain systems. Fig. 6.1 shows the plot of storage efficiency vs. the number of nodes for coded sharding (CS) and PARE-code using constructions 1 and 2. For the plot, we have imposed no lower or upper limit on the storage at each node. For the periodic model we use $(U, D) = [(5, 1), (6, 3), (7, 5)]$. We can clearly see that the PARE-code has a higher storage efficiency compared to coded sharding for different number of nodes in the system. It is also evident that as the number of nodes increases, the improvements in storage efficiency due PARE-code compared to coded sharding increases.

Fig. 6.2 shows the comparison of storage efficiency vs. the number of nodes for different choices of upper and lower limits on the storage at each node. For this figure, we have made use of the same parameters as in Fig 6.1. The curves that have upper and lower limits are indicated by PARE-UL with parameters $\bar{u} = u(n - t_{max}(\mathcal{P}))$ and $\bar{l} = l(n - t_{max}(\mathcal{P}))$. With regards to Lemmas 5 and 6, we have chosen $\bar{u}$ and $\bar{l}$ that satisfy $u \geq \frac{1}{n - t_{max}(\mathcal{P})} \geq l$. For comparison, we also plot the storage efficiency trend for PARE-code with no upper or lower limit (blue plot indicated by PARE) and coded sharding. We can clearly see that the storage efficiency of all the PARE-UL codes is greater than that of coded sharding but less than PARE-codes that have no upper or lower limit on storage. We can also see that as the upper limit is decreased and lower limit is increased, the storage efficiency of PARE-UL codes decreases and moves closer to coded sharding. This is due to the fact that stricter

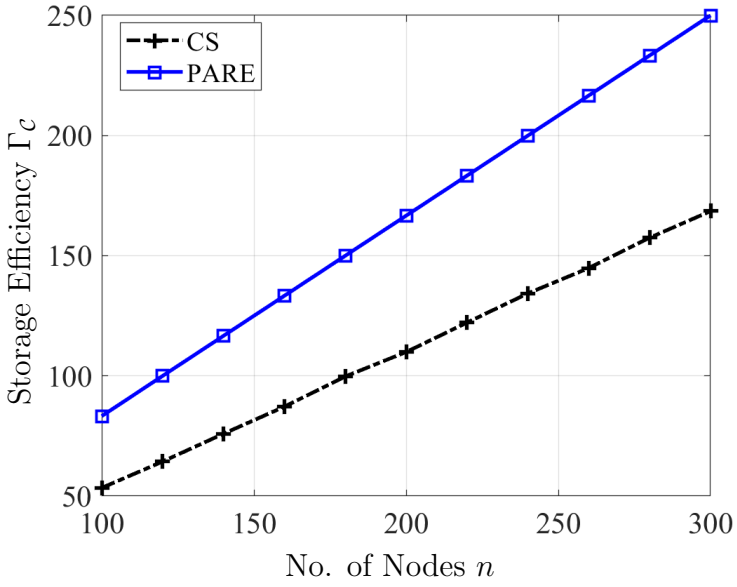


Figure 6.1: Comparison of the storage efficiency vs. number of nodes.

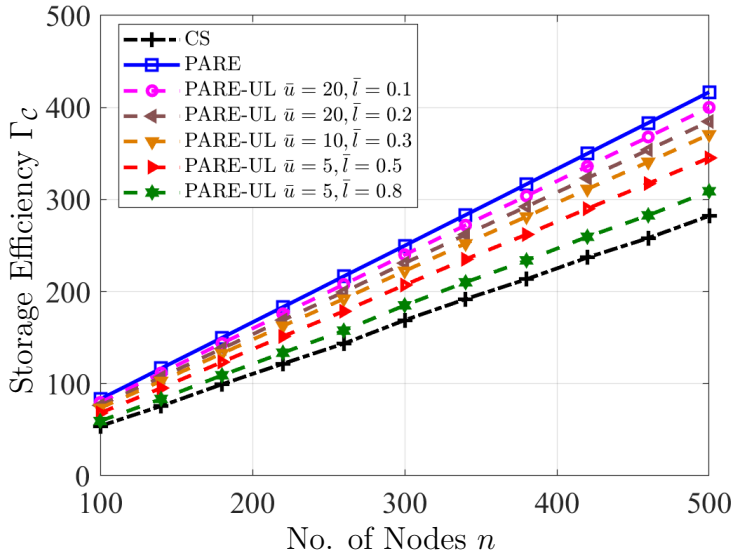


Figure 6.2: Comparison of the storage efficiency vs. number of nodes with different upper and lower storage limits.

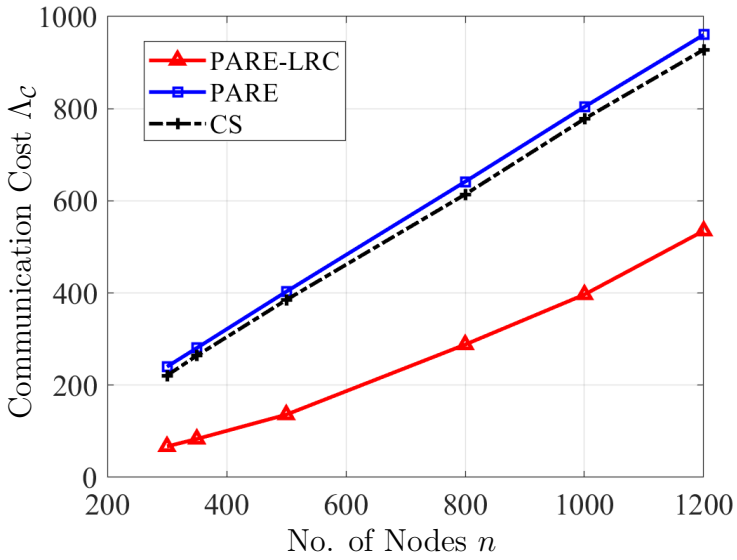


Figure 6.3: Comparison of the communication cost vs. number of nodes with upper and lower storage limits.

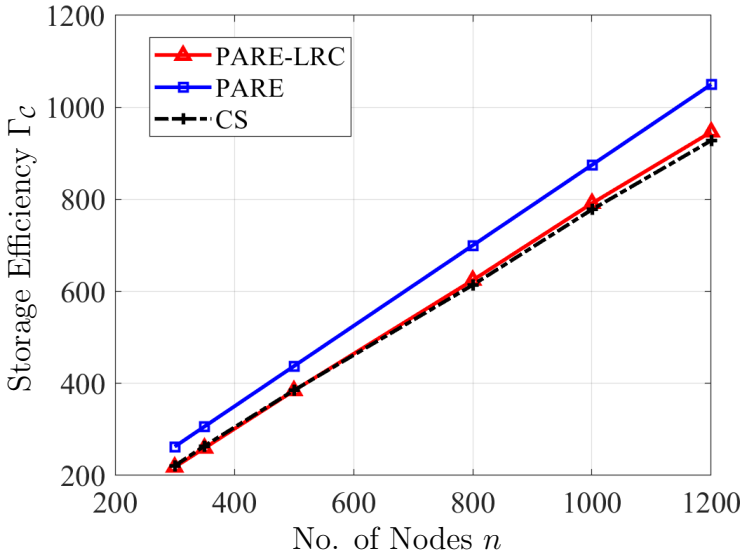


Figure 6.4: Comparison of the storage efficiency vs. number of nodes with upper and lower storage limits.

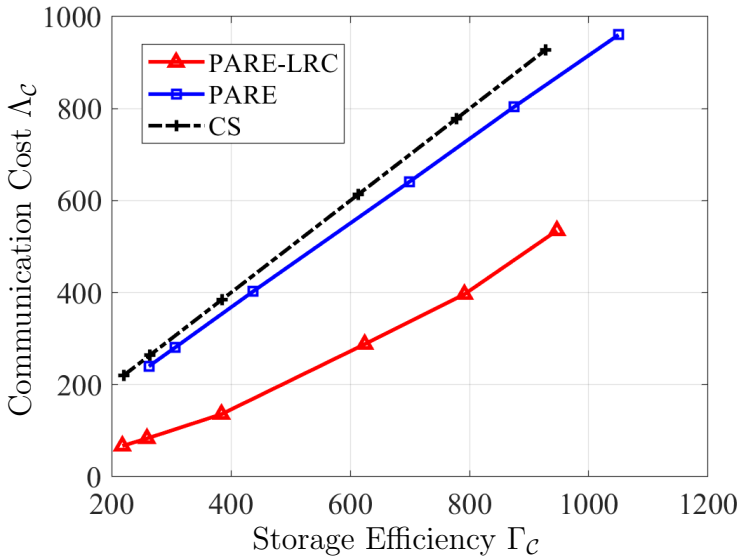


Figure 6.5: Comparison of the storage efficiency vs. number of nodes with upper and lower storage limits.

limits on permissible storage take away the degrees of freedom in allocating shards among the different nodes needed to gain storage efficiency benefits.

Figs. 6.3, 6.4 and 6.5 show the performance of PARE-LRC codes. For these plots, we use the parameters $(U, D) = [(7,1),(6,2),(4,1),(6,1),(3,1)]$, $p = 0.005$, $\bar{l} = 10^{-3}$, $\bar{u} = 20$, $\lambda = 10^{-5}$. In these figures, PARE refers to code constructions 1 and 2 and PARE-LRC refers to code construction 3. As mentioned at the end of Section 4.4, instead of solving Algorithm 1 for $R_{range} = \{1, 2, \ldots, (n - t_{max}(\mathcal{P}))\}$ and $K_{range} = \{(n - t_{max}(\mathcal{P})), \ldots, a(\tilde{r})\}$, we have used the ceiling of eight equally spaced points in $[(n - t_{max}(\mathcal{P})), 2(n - t_{max}(\mathcal{P}))]$ for K_{range} and the ceiling of eight equally spaced points in $[(n - t_{max}(\mathcal{P}))/12, (n - t_{max}(\mathcal{P}))/6]$ as R_{range}. These ranges were heuristically determined to give good performance with respect to $\Lambda_{\mathcal{C}}$ and $\Gamma_{\mathcal{C}}$ and they seem to work well for various choices of (U, D), p, $\bar{u}$ and $\bar{l}$. Since there is a lower limit on storage, from Figs. 6.3 and 6.4 we can see that although PARE-code has a better storage efficiency it has higher communication cost compared to coded sharding. However PARE-LRC code has a lower communication cost and a higher storage efficiency

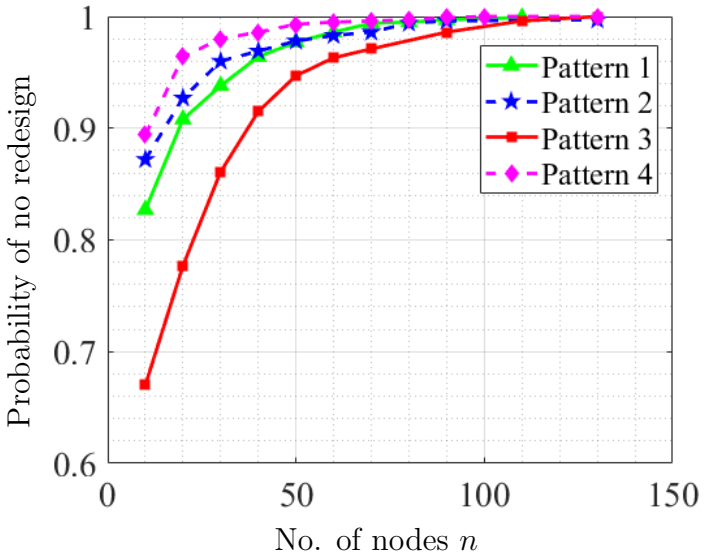


Figure 6.6: Probability of no redesign vs. number of nodes for the case of addition of a node to the system.

compared to coded sharding. It can also be seen from Fig. 6.3 that PARE-LRC code has a much lower communication cost compared to PARE-code and coded sharding. The trade off between communication cost and storage efficiency is shown in Fig. 6.5. Both PARE-code and PARE-LRC code have a better trade off compared to coded sharding. It can also be seen that PARE-LRC code has a much better trade off between communication cost and storage efficiency compared to PARE-code and coded sharding. The trade off is in the sense that, for the same storage efficiency, PARE-LRC codes has the lowest communication cost.

Next we look at the probability of redesign when a new node joins the system as the number of nodes initially in the system increases. This is shown in Fig. 6.6 where we have set no upper or lower limit on storage at each node. The figure depicts the probability that $((\mathrm{x}^{old}, 0), k^{old})$ is an optimal solution after the addition of the $(n+1)^{st}$ node as per Theorem 3. To calculate the probability we run 1000 Monte Carlo trials, and, in each trial, we randomly assign (u_i, d_i, ϕ_i) tuples to each node and check how many of these trials result in $((\mathrm{x}^{old}, 0), k^{old})$ being the optimal solution. We show results for 4 different (U, D) patterns

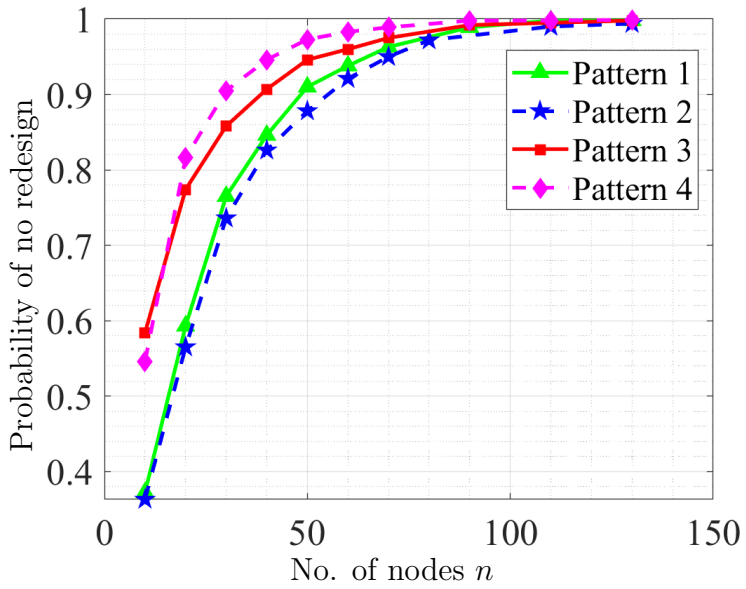


Figure 6.7: Probability of no redesign vs. number of nodes for the case of removal of a node from the system.

in Table 6.1. It is evident that the probability goes to one as the initial number of nodes increases in the system. This shows that when the number of nodes in the system is high, no redesign is typically needed when a new node joins the system.

Fig. 6.7 shows the probability of redesign when a node leaves the system as the number of nodes initially in the system increases. We again set no upper or lower limit on storage at each node. The figure depicts the probability that $(\bar{\mathrm{x}} = (x_1^{old}, \ldots, x_l^{old} + x_n^{old}, \ldots, x_{n-1}^{old}), k^{old})$

Pattern	**(U,D)**
1	[(5,1), (6,3), (7,5)]
2	[(3,1), (2,2), (4,2), (1,1), (5,1), (2,4)]
3	[(11,1), (2,4)]
4	[(8,4), (2,4)]

Table 6.1: Different choices of U and D used in simulation.

is an optimal solution for some l, $1 \leq l \leq (n-1)$, after the n^{th} node (N_n) leaves the system as per Theorem 4. We generate this plot in a manner similar to Fig. 6.6 by running 1000 Monte Carlo trials, randomly assigning (u_i, d_i, ϕ_i) tuples to each node in each trial and checking how many of these trials result in $(\bar{\mathrm{x}} = (x_1^{old}, \ldots, x_l^{old} + x_n^{old}, \ldots, x_{n-1}^{old}), k^{old})$ being the optimal solution for some l, $1 \leq l \leq (n-1)$. We again show results for the (U, D) patterns in Table 6.1. It is evident that for the case when a node leaves the system, the probability goes to one as the initial number of nodes increases in the system. Thus no redesign is typically necessary when a node leaves the system when the number of nodes in the system is high.

CHAPTER 7

Conclusion

In this work we studied the patterned nature of the node failures in blockchains and provided a coding technique called PARE to design codes which have higher storage efficiency compared to the traditionally used coded sharding method. Our code design technique essentially involves solving a linear program which gives us the optimal number of shards the blockchain must be divided into and the optimal number of coded shards which needs to be stored at different nodes. To solve the issue of high communication cost encountered in the above codes, we provided another construction method to design codes that minimally correct only a set of nodes erasure patterns and have an additional property that it can correct all single node failures locally. This construction technique involves solving a series of integer linear programs which again provides the number of shards the blockchain must be divided into and the number of coded shards that needs to be stored at each node. We verify through simulations that these codes have lower communication cost and a better trade off between communication cost and storage efficiency compared to earlier designed PARE-codes and the coded sharding method.

REFERENCES

[AEV16] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman. "MedRec: Using Blockchain for Medical Data Access and Permission Management." In *2016 2nd International Conference on Open and Big Data (OBD)*, pp. 25–30, 2016.

[al18a] D. Perard et al. "Erasure code-based low storage blockchain node." *arXiv:1805.00860*, May 2018.

[al18b] S. Li et al. "PolyShard: Coded Sharding Achieves Linearly Scaling Efficiency and Security Simultaneously." *arXiv:1809.10361*, Sep. 2018.

[Bit20] "Blockchain Luxembourg S.A." In *https://www.blockchain.com/charts/blocks-size, Online*, May 2020.

[BM16] A. Bahga and V. K. Madisetti. "Blockchain platform for industrial internet of things." In *Journal of Software Engineering and Applications*, volume 9, p. 533, 2016.

[CW17] M. J. Casey and P. Wong. "Global supply chains are about to get better, thanks to blockchain." In *Harvard Business Review, [Online]. Available: https://hbr.org/2017/03/global-supply-chains-are-about-to-get-better-thanks-to-blockchain*, Mar. 2017.

[DZW18] M. Dai, S. Zhang, H. Wang, and S. Jin. "A Low Storage Room Requirement Framework for Distributed Ledger in Blockchain." *IEEE Access*, **6**:22970–22975, 2018.

[MD19] D. Mitra and L. Dolecek. "Patterned Erasure Correcting Codes for Low Storage-Overhead Blockchain Systems." In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pp. 1734–1738, 2019.

[Met16] M. Mettler. "Blockchain technology in healthcare: The revolution starts here." In *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pp. 1–3, 2016.

[PKL12] N. Prakash, G. M. Kamath, V. Lalitha, and P. V. Kumar. "Optimal linear codes with a local-error-correction property." In *2012 IEEE International Symposium on Information Theory Proceedings*, pp. 2776–2780, 2012.

[Rip19] "Capacity planning." In *https://developers.ripple.com/capacity-planning.html, Online*, Sep. 2019.

[RMV15] Ankit Singh Rawat, Arya Mazumdar, and Sriram Vishwanath. "Cooperative local repair in distributed storage." *EURASIP Journal on Advances in Signal Processing*, p. 107, 2015.

[RV18] R. K. Raman and L. R. Varshney. "Dynamic distributed storage for scaling blockchains." *arxiv:1711.07617*, Jan. 2018.

[SR19] J. Chung S. Kadhe and K. Ramchandran. "SeF: A Secure Fountain Architecture for Slashing Storage Costs in Blockchains." *arxiv:1906.12140*, Jun. 2019.

[SRK13] N. Silberstein, A. S. Rawat, O. O. Koyluoglu, and S. Vishwanath. "Optimal locally repairable codes via rank-metric codes." In *2013 IEEE International Symposium on Information Theory*, pp. 1819–1823, 2013.

[Swa] M. Swan. "Blockchain: Blueprint for a New Economy." In *Sebastopol, CA: O'Reilly Media, Inc.*

[TR17] N. Teslya and I. Ryabchikov. "Blockchain-based platform architecture for industrial IoT." In *2017 21st Conference of Open Innovations Association (FRUCT)*, pp. 321–329, 2017.

[Wil14] Shawn Wilkinson. "Storj A Peer-to-Peer Cloud Storage Network." 2014.