Doctoral theses at NTNU, 2016:341

Katina Kralevska

Applied Erasure Coding in Networks and Distributed Storage

Doctoral thesis

Katina Kralevska

ISBN 978-82-326-2022-7 (printed ver.)
ISBN 978-82-326-2023-4 (electronic ver.)
ISSN 1503-8181

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology
Thesis for the Degree of
Philosophiae Doctor
Faculty of Information Technology,
Mathematics and Electrical Engineering
Department of Telematics

Doctoral theses at NTNU, 2016:341

NTNU

Katina Kralevska

Applied Erasure Coding in Networks and Distributed Storage

Thesis for the Degree of Philosophiae Doctor

Trondheim, December 2016

Norwegian University of Science and Technology
Faculty of Information Technology, Mathematics
and Electrical Engineering
Department of Telematics

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Information Technology, Mathematics
and Electrical Engineering Department of Telematics

ISBN 978-82-326-2022-7 (printed ver.)
ISBN 978-82-326-2023-4 (electronic ver.)
ISSN 1503-8181

Doctoral theses at NTNU, 2016:341

Printed by NTNU Grafisk senter

Abstract

The amount of digital data is rapidly growing. There is an increasing use of a wide range of computer systems, from mobile devices to large-scale data centers, and important for reliable operation of all computer systems is mitigating the occurrence and the impact of errors in digital data.

The demand for new ultra-fast and highly reliable coding techniques for data at rest and for data in transit is a major research challenge. Reliability is one of the most important design requirements. The simplest way of providing a degree of reliability is by using data replication techniques. However, replication is highly inefficient in terms of capacity utilization. Erasure coding has therefore become a viable alternative to replication since it provides the same level of reliability as replication with significantly less storage overhead.

The present thesis investigates efficient constructions of erasure codes for different applications. Methods from both coding and information theory have been applied to network coding, Optical Packet Switching (OPS) networks and distributed storage systems. The following four issues are addressed:

- Construction of binary and non-binary erasure codes;
- Reduction of the header overhead due to the encoding coefficients in network coding;
- Construction and implementation of new erasure codes for large-scale distributed storage systems that provide savings in the storage and network resources compared to state-of-the-art codes; and
- Provision of a unified view on Quality of Service (QoS) in OPS networks when erasure codes are used, with the focus on Packet Loss Rate (PLR), survivability and secrecy.

A major part of the present thesis is the study of both theoretical and practical aspects of code constructions for distributed storage systems. Distributed storage systems typically employ commodity hardware, often mounted in racks, so that the system can be scaled at a low cost. The components may suffer from failures and other factors, such as software glitches and machine reboots during maintenance operations, that result in unavailability of the stored data. The reliability provided by 3-replication is an accepted industry standard for incorporating reliability into storage systems. Nevertheless, the relentless data growth has made erasure coding a valuable alternative to 3-replication, and hence many distributed storage systems such as Hadoop Distributed File System (HDFS), OpenStack SWIFT and Microsoft Azure employ Reed-Solomon (erasure) codes.

New metrics for efficient erasure coding solutions have been identified in the literature. Some of these metrics, that are also studied in the present thesis, include: 1) reliability, 2) storage efficiency, 3) repair bandwidth, 4) disk-I/O, 5) repair locality, and 6) update complexity. Each of these metrics has a different relevance to a specific system depending on the system's architecture and the workload.

In the present thesis, we propose two novel constructions of erasure codes for distributed storage. The first construction is called *HashTag Erasure Codes (HTECs)*. HTECs are storage-reliability optimal meaning that they offer maximum fault tolerance for the consumed storage. HTECs are the first codes in the literature that reduce the repair bandwidth for both single and multiple failures for an arbitrary sub-packetization level. The bandwidth savings can go up to 70% and 30% compared to RS codes for single and double failures, respectively. HTECs address also the practical problem of disk I/O operations with the focus on reducing the number of random operations that access locations on the storage devices in a non-contiguous manner. The second construction of erasure codes belongs to the class of Locally Repairable Codes (LRCs). The proposed *Balanced Locally Repairable Codes (BLRCs)* are suitable for applications that require a low repair locality for single and double failures, low storage overhead, high reliability and low update complexity.

The present thesis therefore provides new code constructions and demonstrates how these codes are applied to network coding, OPS networks and distributed storage systems.

Preface

This dissertation is submitted in partial fulfillment of the requirements for the degree Philosophiae Doctor (PhD) at NTNU, Norwegian University of Science and Technology. The presented work was carried out at the Department of Telematics (ITEM) in the period October 2012 – August 2016 under the supervision of Associate Professor Harald Øverby and the co-supervision of Professor Danilo Gligoroski and Assistant Professor Gergely Biczók.

Acknowledgement

With deep sense of gratitude, I thank my supervisor Assoc. Prof. Harald Øverby for his guidance and support throughout the PhD period. He has always had an open door and time for me. I would like to thank Prof. Danilo Gligoroski for his constant support and fruitful discussions. The collaboration with him was both inspiring and encouraging. Many thanks to Assis. Prof. Gergely Biczók for his useful advice and insights. I would further like to thank Prof. Steinar Hidle Andresen for accepting me as an IAESTE trainee at ITEM and later supporting me to pursue a PhD.

During my PhD period I have cooperated with the Transfer Technology Office (TTO) at NTNU. I am sincerely grateful to Torbjørn Rostad and the rest of the TTO team who saw a market potential in my research. Per Simonsen, Rune E. Jensen, Sindre B. Stene and Kjetil Babington have contributed greatly in the establishment of the startup company MemoScale by creating a market value and applying the research in industrial products. I would like to thank my co-authors Zoran Hadzi-Velkov, Yanling Chen, Tewelde D. Assefa and Yuming Jiang for the collaboration. Many thanks to my office-mate Chengchen Hu for the useful discussions. I would like to thank Gianfranco Nencioni, Poul E. Heegaard, Katrien De Moor, Mona Nordaune, Randi Fløtnes and the rest of my colleagues who have made ITEM such an enjoyable place to work at. Big thanks to my friends in Trondheim who have always been a great company for going out and exploring new places. Special thanks to Elissar Khloussy for being such a wonderful friend. My dear friends Atina and Jasmina from Macedonia have always been good friends even though we live far away from each other.

My final thanks are reserved to my dearest ones. I would like to thank my parents and brother for supporting me in all my decisions. Special thanks to my grandparents for their constant care. Last but not least, I would like to thank Henry for his understanding and happy moments.

Contents

List of Figures

List of Tables

List of Acronyms

AMDS Almost-Maximum Distance Separable.

ARQ Automatic Repeat reQuest.

BCH Bose-Chaudhuri-Hocquenghem.

BHP Burst Header Packet.

CPT Coded Packet Transport.

CRC Cyclic Redundancy Checksum.

DBR Data Burst Redirection.

DWDM Dense Wavelength Division Multiplexing.

ECC Error-Correcting Code.

FDL Fiber Delay Line.

FEC Forward Error Correction.

HARQ Hybrid Automatic Repeat reQuest.

HDFS Hadoop Distributed File System.

HTECs HashTag Erasure Codes.

LDGM Low-Density Generator Matrix.

LDPC Low Density Parity Check.

LNC Linear Network Coding.

LT Luby Transform.

MBR Minimum-Bandwidth Regenerating.

MDS Maximum Distance Separable.

MRD Maximum Rank Distance.

MSCR Minimum Storage Collaborative Regenerating.

MSR Minimum-Storage Regenerating.

MTFF Mean Time to First Failure.

MTTDL Mean Time To Data Loss.

NLPRS Network Layer Packet Redundancy Scheme.

NMDS Near-Maximum Distance Separable.

Non-MDS Non-Maximum Distance Separable.

OBS Optical Burst Switching.

OPS Optical Packet Switching.

PLR Packet Loss Rate.

PM-MSR Product-Matrix-MSR.

PRNG Pseudo-Random Number Generator.

QoS Quality of Service.

RAID Redundant Arrays of Inexpensive Disks.

RAM Random-Access Memory.

RLNC Random Linear Network Coding.

RS Reed Solomon.

RSA Rivest-Shamir-Adleman.

RTT Round Trip Time.

SECDED Single-Error-Correcting and Double-Error-Detecting.

SHEC Shingled erasure codes.

SSAC Small Set of Allowed Coefficients.

WRON Wavelength Routed Optical Networks.

Part I

Summary of the Thesis

Chapter 1
Introduction

1.1 Thesis Structure

The present thesis is a collection of papers which is in accordance with NTNU rules for PhD studies. It is divided into two main parts:

- **Part I: Summary of the Thesis**
- **Part II: Included Papers**

Part I is a comprehensive summary of the present thesis. It consists of three chapters:

- The *Introduction* chapter (Chapter 1) presents the motivation for applying erasure codes in different networks.
- The *Background and Related Works* chapter (Chapter 2) gives the necessary background to understand the contributions of the thesis. It also reviews the state-of-the-art for erasure coding, algorithms for header compression in network coding, code constructions for distributed storage and QoS metrics in OPS networks. Some of the challenges faced by the coding and the networking communities that are addressed in the present thesis are listed in the end of each section.
- The *Contributions and Concluding Remarks* chapter (Chapter 3) presents the research questions answered in the present thesis and the research contributions and results obtained during the PhD period. It also puts the presented research results into a wider context by comparing them to selected references. Finally, conclusions followed by suggestions for future work are presented.

Part II consists of 7 papers where 6 are published and 1 is submitted for publication (Table 3.1).

1.2 Motivation

The relentless data growth brings enormous challenges, as well as incredible research and business opportunities. IDC [IDC12] estimates that the total amount of digital data created, replicated and consumed will reach 40000 exabytes, or 40 trillion gigabytes, by 2020. From now until 2020, the amount of digital data is expected to double every two years, as shown in Figure 1.1.

Reliable communication through unreliable media is paramount in modern communication systems. Reliable communication requires that all intended receivers of the data receive the data intact, i.e. data must be transferred without errors or loss. The demand for efficient and reliable communication has been accelerated even more by the emergence of large-scale, high-speed data networks for exchange, processing and storage of digital information in public and private spheres. Reliability is achieved by adding redundancy at different levels in the protocol stack. One way is to use Forward Error Correction (FEC) codes as error correction or erasure codes. FEC codes preprocess the data in such a way to provide recovery after data corruption [CC81].

Erasure coding has emerged as a compliment, or an alternative, to Automatic Repeat reQuest (ARQ) and replication in communication networks and distributed storage systems, respectively.

Using erasure coding, or combining it with ARQ, is a better solution instead of only using ARQ. When the PLR is very high, then retransmissions happen frequently and the system throughput is reduced significantly. In this case, combining ARQ with FEC known as Hybrid Automatic Repeat reQuest (HARQ) [CC84, LY82] is useful. ARQ is not feasible for unidirectional broadcast networks or real-time applications, because a return channel may not exist or the Round Trip Time (RTT) delay may be too large. Additionally, when the number of users is very large, scalability issues may prevent the use of return channels. In all these scenarios, the use of erasure codes is imminent.

Replication is not an appropriate enabler of the exabyte era because it increases enormously the storage overhead. Let us take the following example where 3-replication and an $(9, 6)$ erasure code provide a similar level of reliability. Three copies of the same file are stored with 3-replication, while the file is divided into 6 fragments and 3 redundancy fragments when constructed with an $(9, 6)$ erasure code. The storage overhead is 200% with 3-replication, while it is only 50% with the erasure code. In this example, erasure coding reduces the cost of storage by 150%, which is a tremendous cost saving when storing exabytes of data. Erasure coding has a clear advantage over replication as it provides the same level of reliability with less storage overhead [WK02].

Although many different erasure codes have been developed, there is no erasure code construction that provides simultaneously optimal performance and reliability. Maximum Distance Separable (MDS) codes such as Reed-Solomon codes [RS60] are

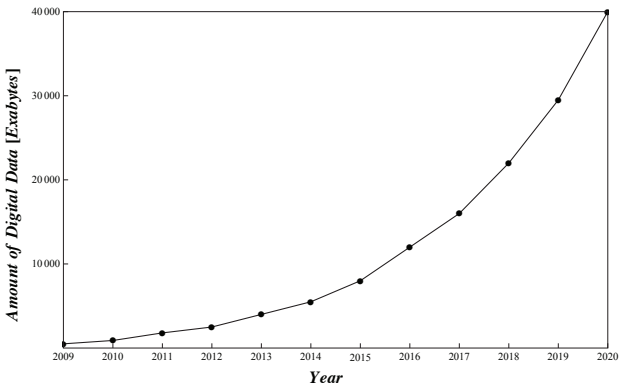


Figure 1.1: 50-fold growth of the amount of digital data from 2010 to 2020 [IDC12].

fault tolerant optimal, but they are computationally expensive in practice. Low Density Parity Check (LDPC) codes [Gal63] do not offer the same fault tolerance as MDS codes, but tend to be computationally inexpensive and may have regular or irregular fault tolerance. Random Linear Network Coding (RLNC) is a flexible coding scheme that does not follow a predesigned code. Depending on the finite field size, the codes can be fault tolerant optimal. Then the codes are computationally demanding as Reed-Solomon codes.

Both the underlying system and the application have a huge impact on which coding scheme gives the best performance. For instance, erasure coding has been currently deployed in Windows Azure [HSX+12], big data analytics clusters (e.g. Facebook Analytics Hadoop cluster [SAP+13]), archival storage systems, and peer-to-peer storage systems like Cleversafe. Facebook has recently reported that it is archiving old data using a classical Reed-Solomon erasure code implemented on the top of HDFS[SAP+13], while Microsoft uses a pyramid code as the main storage primitive of its Azure storage service [HCL13]. Reed-Solomon codes, that are the essential building blocks in RAID 6, are optimal in terms of storage capacity utilization in large-scale distributed storage systems, but they perform poorly in terms of other system resources such as disk I/O and network bandwidth. During a recovery of lost or otherwise unavailable data, classical Reed-Solomon codes require a large amount of data to be read and transferred across the network.

Accordingly, the work presented in this thesis has been directed towards constructing efficient erasure codes for different applications. The four topics covered in the present thesis are:

- Construction of binary erasure codes (Paper 1 and Paper 2);
- Reduction of the header overhead due to coding coefficients in network coding (Paper 3);

- Construction of erasure codes for practical use in distributed storage systems (Paper 4, Paper 5 and Paper 6); and
- Application of erasure codes to increase the QoS in OPS/OBS networks (Paper 7).

Chapter 2
Background and Related Works

In this Chapter we review the background and the related works in the main research areas of this dissertation: erasure coding (Section 2.1), header compression algorithms for network coding (Section 2.2), code constructions for distributed storage systems (Section 2.3) and optical packet switched networks (Section 2.4).

2.1 Coding Theory

Data reliability, regardless of the medium, is achieved by adding redundancy. FEC (Forward Error Correction) codes are either used as *error correction* or *erasure codes*. Error correction codes are usually used at lower layers of the protocol stack, either as standalone codes or in conjunction with error detection checksums (e.g. Cyclic Redundancy Checksum (CRC))[PB61]. The upper layers deal with erasures (missing data units) after reception or storage/retrieval. Erasure codes are typically used in situations where the exact positions of the missing data units are known *a priori* (e.g. disk array). In many cases, error correction and erasure codes use the same encoding algorithm but have different decoding algorithms.

Since FEC codes treat each symbol as an element of a finite field and they perform extensively operations in finite fields, i.e. Galois fields, first a very brief introduction to Galois fields is given. For a more detailed description of Galois fields, please refer to [LN86].

2.1.1 Galois fields

Finite fields $\mathbf{F}_q$, also known as Galois fields $GF(q)$, are fundamental to coding theory. We use both notations $\mathbf{F}_q$ and $GF(q)$ interchangeably through the present thesis. The main advantage of using a Galois field is its closure property. A field is closed under both addition and multiplication meaning that the result of addition and multiplication of field elements is still a field element. Both operations are associative, distributive and commutative. Every non-zero element has a multiplicative inverse and every element has an inverse additive (negative) element. Working with data

in transit or with data at rest in a Galois field means that the data elements are mapped into field elements, the performed operations follow the rules of the field and the data is reconstructed by inverse mapping.

The *order* of $GF(q)$ is the number of elements in the field. There exists a finite field of order q if, and only if, q is a prime power, i.e. $q = p^m$ where p is a *prime number* called the *characteristic* of the field, and m is a positive integer. If $m = 1$, then the field is called a *prime field*. Working in a prime field $GF(p)$ is quite simple. The prime field is the set of elements from 0 to $p-1$ under the operations of addition and multiplication modulo p.

If $m \geq 2$, then the field is called an *extension field*. Galois fields of order 2^m are called *binary extension fields*. These fields are used ubiquitously in coding theory and cryptography. One way of representing the elements in $GF(2^m)$ is to use a set of polynomials of degree at most $m-1$ where the coefficients are from the binary field $GF(2) = \{0, 1\}$:

$$GF(2^m) = \{c_{m-1}x^{m-1} + c_{m-2}x^{m-2} + \ldots + c_1x^1 + c_0x^0 : c_i \in \{0, 1\}\}. \tag{2.1}$$

Thus, the 4-bit element $a = (1, 1, 1, 0)_{2^4}$ has the following polynomial representation $a(x) = 1x^3 + 1x^2 + 1x + 0$.

The field $GF(2^m)$ is defined over an *irreducible polynomial* $f(x)$ of degree m with coefficients from $GF(2)$. An irreducible polynomial is analogous to a prime number in that it cannot be factored as a product of polynomials each of degree less than m. Addition in $GF(2)$ is done with the bitwise XOR operator, while multiplication is performed with the bitwise AND operator. Addition in $GF(2^m)$ is the usual addition of polynomials with coefficient arithmetic performed modulo 2, while multiplication is more complex. Multiplication of two field elements a and b is performed by polynomial multiplication of $a(x)$ and $b(x)$ and then the product $a(x)b(x)$ is reduced modulo the irreducible polynomial $f(x)$.

Another useful property of any finite field $GF(q)$ is that the set of all nonzero elements $GF(q)^\times = GF(q) \setminus 0$ form a *multiplicative cyclic group* $(GF(q)^\times, \times)$. That means that any nonzero element $a \in GF(q)^\times$ can be represented as a power of a single element $\alpha \in GF(q)^\times$. Such a generator α is called a *primitive element* of the finite field. Powers of α repeat with a period length of $q-1$, thus, $\alpha^{q-1} = \alpha^0 = 1$. This makes multiplication of two elements $a = \alpha^i$ and $b = \alpha^j$ quite simple and fast. If we write $i = \log(a)$ and $a = \text{antilog}(i)$, then the product of $a, b \in GF(2^m)$ is computed as $ab = \text{antilog}(\log(a) + \log(b)) \mod 2^m - 1$.

2.1.2 Erasure Coding

Erasure codes, in particular linear block codes, are most appropriate for the applications such as data transmission and storage, which are of interest to the present thesis. The background presented in this subsection is essential to fully understand

the work in the present thesis. Some terminology and definitions can be found in [LC83, Pla05].

A block encoder, according to certain rules, transforms each input message of k information symbols into a message of n symbols ($n > k$). This message of n symbols is called a *codeword*. If the alphabet of the information source contains q different digits, then there are a total of q^k distinct messages of k symbols. To each of the q^k possible messages a unique codeword is assigned. This set of q^k codewords of n symbols is called a *q-ary block code of length n*.

Definition 2.1. A block code of length n and q^k codewords is called a *linear (n,k) code* if and only if its q^k codewords form a k-dimensional subspace of the vector space of all n-tuples over the finite field $GF(q)$.

The *code rate* R, or code efficiency, is defined as $R = \frac{k}{n}$. The error detection and error correction capabilities of a (n, k, d) code are defined by its metric, the minimum distance d. Before defining d, it is first necessary to define the Hamming weight of a codeword and the Hamming distance between two codewords.

Definition 2.2. The *Hamming weight* $w_H(c)$ of a codeword c is the number of non-zero elements in c.

Definition 2.3. The *Hamming distance* $d_H(c_1, c_2)$ between two codewords c_1 and c_2, that have the same number of elements, is the number of elements in which these two codewords differ.

Consider the two codewords $c_1 = (0, 1, 0, 0, 1)$ and $c_2 = (1, 1, 0, 1, 1)$. The Hamming weights of c_1 and c_2 are $w_H(c_1) = 2$ and $w_H(c_2) = 4$, respectively, while $d_H(c_1, c_2) = 2$.

Definition 2.4. The *minimum distance* of a (n, k) block code C, denoted by d, is the minimum among the Hamming distances between any two different codewords from C. A block code C of length n, q^k codewords and minimum distance d is denoted by (n, k, d) code C.

Theorem 2.5. *[Sin64] For a $(n, k, d)_q$ linear code we have*

$$d \leq n - k + 1 \quad \text{(Singleton bound)}. \tag{2.2}$$

Codes for which the equality holds are known as Maximum Distance Separable (MDS) codes.

When MDS codes are used as erasure codes, the receiver can recover the k source symbols from any subset of k received symbols out of the n encoded symbols.

The *Singleton defect* of a (n, k, d) code C, that is defined as $s(C) = n - k + 1 - d$, measures how far away is C from being a MDS code. Based on the Singleton defect, the codes are divided in two classes:

1. Optimal, or very close to optimal, ones known as MDS [MS78], Almost-Maximum Distance Separable (AMDS) [dB96] and Near-Maximum Distance Separable (NMDS) [DL95].

2. Suboptimal or Non-Maximum Distance Separable (Non-MDS) codes [GLW10, Haf05, HCL13, LMS$^+$97].

Definition 2.6. A (n, k, d) code C is
(i) *t-error detecting code* iff its minimum distance is at least $t + 1$;
(ii) *t-error correcting code* iff its minimum distance is at least $2t + 1$.

As mentioned earlier, a linear code constitutes a subspace and thus any codeword can be represented by a linear combination of the basis vectors of the subspace, i.e. by a linear combination of linearly independent vectors. The basis vectors can be written as rows of a $k \times n$ matrix.

Definition 2.7. Any matrix $\mathbf{G} \in \mathbf{F}_q^{k\times n}$ whose rows form a basis of C is called a *generator matrix* for C, i.e.

$$C = \{\mathbf{xG} \in \mathbf{F}_q^n : \mathbf{x} \in \mathbf{F}_q^k\} = \{\mathbf{y} \in \mathbf{F}_q^n\}. \tag{2.3}$$

The injective map $\mathbf{F}_q^k \rightarrowtail \mathbf{F}_q^n$, $\mathbf{x} \rightarrowtail \mathbf{xG}$ encodes all the q-ary words of length k to words of length n.

A generator matrix is called *systematic* if it is of the form

$$\mathbf{G} = \Big[I_k|P\Big], \tag{2.4}$$

where I_k is an identity matrix of order k and P is a $k \times (n - k)$ parity submatrix.

The code can either be systematic or non-systematic. The generator matrix of a systematic code has the same form as the matrix represented in (2.4). That means that a systematic linear code does not transform the original k data symbols, but it only generates extra r redundant symbols. If the first k rows in $\mathbf{G}$ do not contain an identity matrix, then the code is non-systematic. That is to say, all n generated symbols linearly depend on all original k symbols via $\mathbf{G}$. Systematic codes are less computationally demanding than non-systematic, since they do not require processing of the original data.

Given a generator matrix $\mathbf{G}$ of a linear code we can derive its parity-check matrix $\mathbf{H}$ (and vice versa).

Definition 2.8. The *parity-check matrix* $\mathbf{H}$ for a (n, k, d) linear code with a generator matrix $\mathbf{G}$ (2.4) is given by

$$\mathbf{H} = \Big[-P^T|I_{n-k}\Big], \tag{2.5}$$

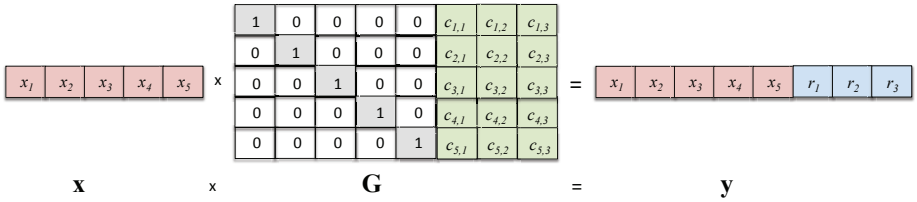


Figure 2.1: Encoding of the source data $\mathbf{x}$ with the generator matrix $\mathbf{G}$ of a $(8, 5)$ MDS code.

Since the rows of $\mathbf{H}$ are linearly independent, it generates a $(n, n-k, d')$ linear code called the *dual code* of the (n, k, d) linear code generated with $\mathbf{G}$.

Codes with generator matrices that are sparse and balanced minimize the maximal computation time of computing any code symbol. The problem of constructing balanced and sparse codes was studied in [DSDY13, HLH16].

Theorem 2.9. *[DSDY13] Suppose* $1 \leq k \leq n$ *and* $q > \binom{n-1}{k-1}$. *Then there always exits a* $(n, k)_q$ *MDS code that has a generator matrix* $\mathbf{G}$ *satisfying the following two conditions:*

- *Sparsest: each row of* $\mathbf{G}$ *has weight* $n - k + 1$*; and*
- *Balanced: column weights of* $\mathbf{G}$ *differ from each other by at most one.*

Let us explain the encoding and decoding of data with a $(8, 5)$ MDS code. As shown in Figure 2.1, $k = 5$ source data units represented as a row vector $\mathbf{x}$ are encoded into $n = 8$ data units (a row vector $\mathbf{y}$) with the systematic generator matrix $\mathbf{G}$ of a $(8, 5)$ MDS code. The coefficients $c_{i,j}$, $i = 1, \ldots, 5$ and $j = 1, 2, 3$, are elements from $GF(q)$. The encoder in Figure 2.2 performs this multiplication. Next, the data is transmitted through unreliable medium to the receiver. Some of the data units may get corrupted (lost) during the transmission. Since the code is MDS, at most 3 data units can be lost. In this example, x_1, x_4, r_2 are lost. Namely, the decoder has to receive at least 5 data units in order to reconstruct the source data. Recovery of the source data is done by

$$\mathbf{y}' = \mathbf{x}\mathbf{G}' \rightarrow \mathbf{x} = \mathbf{y}'\mathbf{G}'^{-1}, \tag{2.6}$$

where $\mathbf{x}$ is the source data and $\mathbf{y}'$ is the subset of k components available at the decoder. The matrix $\mathbf{G}'$ is the subset of columns from $\mathbf{G}$ corresponding to the components of $\mathbf{y}'$. The source data $\mathbf{x}$ can be reconstructed only if $\mathbf{G}'$ is non-singular. This means, in general cases, any $k \times k$ submatrix extracted from $\mathbf{G}$ has to be invertible in order to recover from at most $n - k$ lost data units. In the presented example, the matrix $\mathbf{G}'$ is obtained by deleting the columns in $\mathbf{G}$ that correspond to

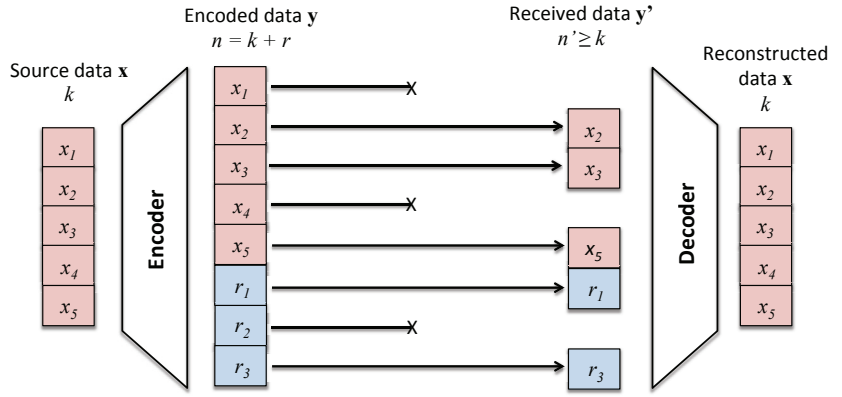


Figure 2.2: A graphical representation of the encoding/decoding process. The encoder encodes k source units into n. The decoder has to receive at least k data units in order to reconstruct the source data.

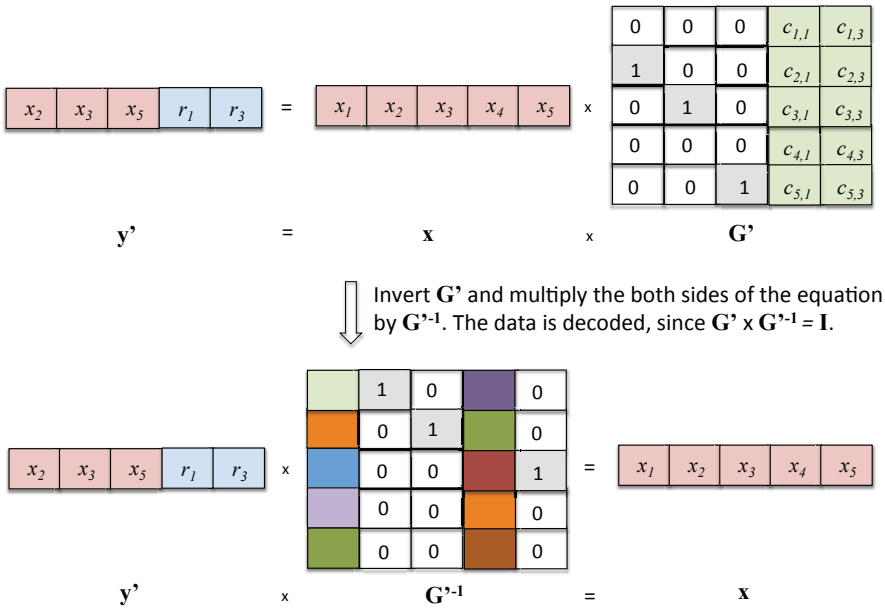


Figure 2.3: A graphical representation of the decoding process at the decoder.

the lost data units. Decoding is done in two steps: first $\mathbf{G}'$ is inverted and then the data is decoded by computing $\mathbf{x} = \mathbf{y}'\mathbf{G}'^{-1}$ as shown in Figure 2.3.

There are numerous types of erasure codes. Some of the codes, such as Luby Transform (LT) [Lub02], Tornado [LMSS01a], and Raptor [Sho06] are protected by patents and hence their further development by third parties is problematic.

Among all codes in the class of block codes, cyclic codes are the most important from practical point of view. Bose-Chaudhuri-Hocquenghem (BCH) codes were discovered independently in the papers by Bose and Chaudhuri (1960) [BRC60] and Hocquenghem (1959) [Hoc59]. BCH codes are cyclic codes that have algebraic decoding algorithms.

Reed Solomon (RS) codes were first described in a paper by Reed and Solomon in 1960 [RS60]. They are non-binary BCH codes defined by the parameters: $n = q - 1, n - k = 2t, d = 2t + 1$. Since the minimum distance is $n - k + 1$, RS codes are MDS codes. RS codes are widely implemented in storage devices and communication standards. RS encoding is relatively straightforward, but decoding is complex despite the significant efficiency improvements with Berlekamp-Massey algorithm [Ber68]. The two limitations of RS codes are: the small block size and the high decoding times. The length of a RS code is limited by the field size, for example, it is $n \leq 255$ for $GF(256)$. The larger Galois field is, the longer the code can be, but at the same time the operations are getting slower and more complex. Therefore, the values of n and k have to be small if high transmission rates are desired.

The introduction of Turbo codes [BGT93] and LDPC codes [Gal63, MN97, MN95] have been one of the most important milestones in channel coding during the last years. Provided that the information block length is long enough, both Turbo and LDPC codes achieve performance close to the Shannon theoretical limit. However, in practical applications, both schemes have some complexity issues. Specifically, the encoding complexity is very low, but the decoding is more complex with Turbo codes compared to LDPC codes. The contrary occurs for standard LDPC codes. The encoding is more complex with standard LDPC codes, but the decoding is simpler.

LDPC codes were first introduced by Gallager in 1960 [Gal63], but they were impractical for implementation in that time. As the computational power has increased, they become attractive for research and practical implementations. MacKay and Neal rediscovered them in 1995 [MN95]. LDPC codes are defined by sparse parity-check matrices. Sparse bipartite graphs are used to represent LDPC codes where one set of nodes, the variable nodes, corresponds to elements of the codeword and the other set of nodes, the check nodes, corresponds to the set of parity-check constraints which define the code. An example of a regular LDPC code where all nodes of the same type have the same degree is shown in Figure 2.4. The principle of using irregular graphs where the degrees of the variable and the check nodes can vary widely was introduced in [LMS+97], and it was further studied in [LMSS01b, RSU01]. The degrees of each set of nodes are chosen according to some distribution. The

decoding complexity for LDPC codes increases linearly with the block length. LDPC codes are asymptotically good and their recovery performance decreases for small blocks.

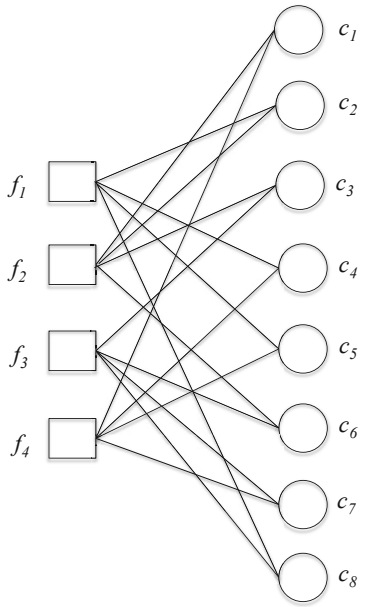


Figure 2.4: A $(2, 4)$ regular LDPC code where $k = 8$ and $r = 4$. All 8 variable nodes have degree 2 and all 4 check nodes have degree 4.

Garcia-Frias and Zhong introduced Low-Density Generator Matrix (LDGM) codes that are constructed by using systematic sparse generator matrices [GFZ03]. They are a special class of LDPC codes with low encoding and decoding complexity. The amount of processing at the encoder is comparable to that of Turbo codes due to the sparseness of the generator matrix.

LDPC were significantly improved by Luby, Shokrollahi et al. that led to the invention of Tornado [BLMR98], LT [Lub02] and Raptor [Sho06] codes. Tornado codes are the precursor to fountain codes. Fountain or rateless codes are a class of erasure codes with the property that a potentially limitless sequence of encoding symbols can be generated from a given set of source symbols such that the original source symbols can ideally be recovered from any subset of the encoding symbols of size equal to or only slightly larger than the number of source symbols. These codes do not have a fixed code rate and are known as rateless codes. Both LT and Raptor codes belong to the class of rateless codes.

The fundamental goal of the research presented in the present thesis is to construct erasure codes that have the following desirable properties:

- High code rate;
- Same error-correcting capabilities as MDS codes, or very close to those of MDS codes; and
- Efficient encoding and decoding algorithms.

2.2 Network Coding

Network coding as a research area was initiated by the seminal paper by Ahlswede et al. [ACLY00]. They made the key observation that intermediate nodes are allowed to carry out algebraic operations on the incoming data instead of only forwarding the incoming data. Before defining their main result, some essential terminology that can be found in [MS12, Law01, Bol79, RSK10] is introduced.

A communication network is defined as a tuple $N = (V, E, S, T)$ that consists of:

- a finite directed acyclic multigraph $G = (V, E)$ where V is the set of vertices and E is the multiset of directed edge;
- a set $S \subset V$ of sources; and
- a set $T \subset V$ of sink nodes.

Vertices model communication nodes within the network, while directed edges model error-free communication channels between the nodes. An edge (i, j) has unit capacity in the sense that it can be used to reliably deliver one symbol from i to j. To allow for greater capacity from i to j, multiple edges between i and j are permitted, i.e. G is in general a multigraph. The capacity of an edge $(i, j) \in E$ is given by R_{ij} and let $R = [R_{ij}, (i, j) \in E]$.

Definition 2.10. $F = [F_{ij}, (i, j) \in E]$ is a *flow* in G from s where $s \in S$ to t_l where $t_l \in T$ if for all $(i, j) \in E$

$$0 \leq F_{ij} \leq R_{ij}$$

such that for all $i \in V$ except for s and t_l

$$\sum_{i':(i',i)\in E} F_{i'i} = \sum_{j:(i,j)\in E} F_{ij},$$

i.e. the total flow into node i is equal to the total flow out of node i.

F_{ij} is referred to as the value of F in the edge (i, j). The value of F is defined as

$$\sum_{j:(s,j)\in E} F_{sj} - \sum_{i:(i,s)\in E} F_{is}$$

which is equal to

$$\sum_{i:(i,t_l)\in E} F_{it_l} - \sum_{j:(t_l,j)\in E} F_{t_l j}.$$

Definition 2.11. F is a *max-flow* from s to t_l in G if F is a flow from s to t_l whose value is greater than or equal to any other flow from s to t_l.

Definition 2.12. A *cut* is a set of edges that partition the graph into two sets of vertices.

Definition 2.13. A *minimal cut* separating s and t is a cut of the smallest cardinality denoted as min-cut(s, t).

A min-cut(s, t) can be regarded as a bottleneck between s and t and it limits the information rate of the flow between s and t.

Theorem 2.14. *[Law01]**Max-Flow Min-Cut Theorem:** For every non-source node t, the minimum value of a cut between the source and a node t is equal to* max-flow*(s, t).*

It is well known from Menger's Theorem [Men] that the number of edge-disjoint paths from s to t is equal to max-flow. A collection of edge-disjoint paths can be found by the Ford-Fulkerson algorithm [FF]. Thus, in a single-sink network, the number of symbols transferred from s to t per time unit is equal to the min-cut of the network where each symbol is sent on a different edge-disjoint path.

Ahlswede et al. [ACLY00] showed that the multicast capacity for a single-source network, i.e. the maximum rate at which s can transfer information to the sinks, cannot exceed the capacity of any cut separating s from the sinks.

Theorem 2.15. *[MS12] The multicast rate $R(s, T)$ from s to T cannot exceed the transmission rate that can be achieved from s to any element of T. The multicast rate $R(s, T)$ must satisfy:*

$$R(s, T) \leq \min_{t \in T} \text{min-cut}(s, t). \tag{2.7}$$

The quantity $\min_{t \in T} \text{min-cut}(s, t)$ is referred to as the multicast capacity of the given network. The upper bound is achievable (with equality) via network coding [ACLY00].

Li et al. [LYC03] showed that Linear Network Coding (LNC) achieves the upper bound given that the finite field is sufficiently large.

Theorem 2.16. *Adapted version of [LYC03, Th.5.1] from [RSK10]: Let q be a sufficient large power of 2. A symbol over $\mathbf{F}_q$ is treated as a unit of information. In a directed, delay-free, acyclic graph, with a single source s and multiple sinks $t_1, \ldots, t_k$ and where the edges have integral capacity, if the capacity of the min-cut from the source to each of the k sinks is at least ν, then there exists a linear network solution that delivers ν units of information to each of the k sinks simultaneously.*

Koetter and Médard extended further the work by Li et al. [LYC03] to arbitrary networks and robust networking. In [KM03], they presented an algebraic framework for investigating coding rate regions in networks using linear codes.

An efficient distributed randomized approach that asymptotically achieves the capacity for general multi-source multicast networks is presented in [HMK+06]. Random Linear Network Coding (RLNC) is a simple, randomized coding method that maintains "a vector of coefficients for each of the source processes," which is "updated by each coding node". In other words, RLNC requires messages being communicated through the network to be accompanied by some degree of extra information (a vector of coefficients). The vector of coefficients is updated at each node that performs network coding.

Another definition of network coding is coding at a node in a packet network (where data is split into packets and network coding is applied to the content of packets). We use this definition in the sequel.

A well-known benefit of network coding is an increase of the throughput. This is achieved by using packet transmissions more efficiently, i.e. by communicating more information with fewer packet transmissions. The famous butterfly network in Figure 2.5 illustrates this. Assume that the source node s wants to multicast two packets to the destinations t_1 and t_2. Assume that the capacity of each link is 1 packet per time unit and the delay of each link is the same. The maximum throughput from the source node s to the destination nodes t_1 and t_2 is 2 packets per time unit, but the maximum throughput cannot be achieved simultaneously if only routing is allowed, since node n_3 can transmit either b_1 or b_2 but not both packets at the same time. Nevertheless, if node n_3 performs the exclusive-OR (XOR) operation on b_1 and b_2 and transmits the XOR-ed packet to node n_4, then both destinations achieve the maximum throughput simultaneously. Node t_1 decodes correctly b_2 after it receives b_1 from node n_1 and the XOR-ed packet from node n_4. It is similar for node t_2.

At the expense of encoding operations at the intermediate nodes and decoding operations at the sink nodes, RLNC improves the network throughput, the efficiency and the scalability, as well as the resilience to attacks and eavesdropping [CY02, BN05]. Inspired by these gains, researchers have applied network coding in many applications such as wireless networks [KRH+08, KHH+13], distributed storage systems [ADMK05, DGW+10], video streaming [NNC10], satellite networks [VB09] and distributed file sharing [WWX10, FR12, GR06].

However, there are some issues with practical implementation of network coding. In order to explain them easily, the generation of a coded packet y_k in RLNC is presented in Figure 2.6. The file is divided into n packets of length l and encoding is performed within a group of m packets. This group is called a generation and m is the generation size. Random coefficients are generated and each packet is multiplied with a coefficient. Then, all packets are XOR-ed together, i.e. bitwise XOR-ing of packets with equal length, and y_k is generated. The newly generated packet is

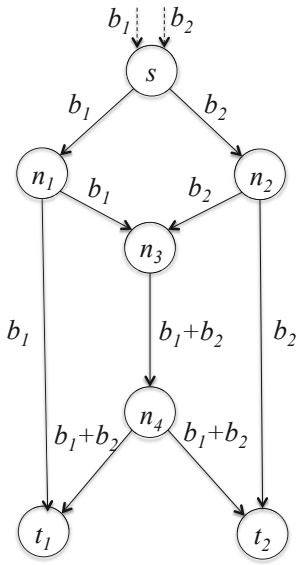


Figure 2.5: Butterfly network

a linear combination of m packets from the generation and each newly generated packet should be linearly independent from previously generated packets of the same generation. The average number of packets that have to be received before the original m packets can be decoded is upper bounded by [LMS09]

$$\min\left\{m\frac{q}{q-1}, m+1+\frac{1-q^{-m+1}}{q-1}\right\}. \tag{2.8}$$

The exact probability of successful decoding is derived in [TCBOF11]. The probability of generating a linearly independent packet increases with the number of packets in the generation m and the size of the Galois field q. On the other hand, the length of the header overhead due to the coding coefficients becomes significant. This affects the throughput of a system and has a huge impact on the system load for some network scenarios. Thus, it is of a great importance to find a good tradeoff between the parameters.

Next, we calculate the encoding complexity of RLNC. The computational complexity of generating the coding coefficients depends on the complexity of generating a random number r that is system specific and the generation size m. Consequently, the complexity of generating the encoding coefficients is $\mathcal{O}(mr)$ where r is a constant. After generating the encoding coefficients, the packets are multiplied with them. This has a complexity of $\mathcal{O}(ml)$. Finally, the complexity of encoding one packet y_k is $\mathcal{O}(m(l+r))$ [HPFL08], while encoding all packets within one generation has

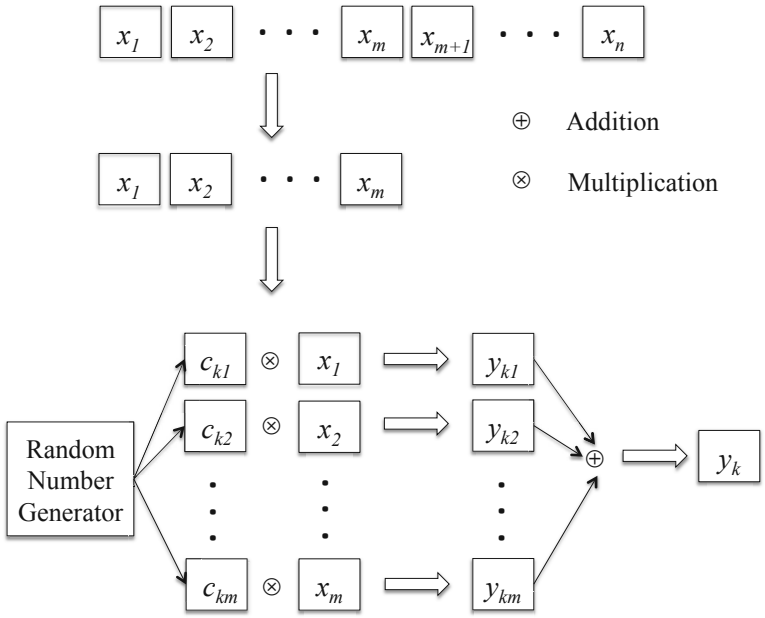


Figure 2.6: Generating a coded packet y_k in RLNC. The file is split into n packets and encoding is performed within a group of $m < n$ packets. Each packet is multiplied with a random coefficient c_{k_i}. All packets are XOR-ed together and y_k is generated.

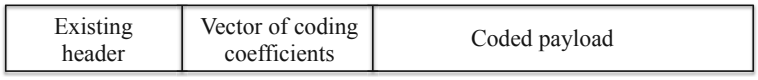


Figure 2.7: Structure of a coded packet in RLNC.

a complexity equal to $\mathcal{O}(m^2(l+r))$. The structure of a coded packet in RLNC is showed in Figure 2.7. The length of the vector of coding coefficients is $m \log_2 q$ bits.

For instance, a typical packet length in sensor networks is 30 bytes. Consider a sensor network where 60 nodes send data. If RLNC in $GF(16)$ is performed and the generation size is 60, then 30 bytes per packet are used for recording the coding coefficients, i.e. the length of the header overhead is equal to the length of the useful data. Additionally, the header overhead has an impact on the required energy to transmit the coded packets. The energy used to transmit a single bit of data between devices in ad hoc sensor networks is equal to the energy for performing 800 instructions in the devices [MFHH02]. This implies that many applications may benefit by performing local computations rather than sending more bits. Thus, the reduction of the length of transmitted data, while keeping the same functionality of the employed algorithms is a challenging task.

Several header compression algorithms have been suggested in recent literature. Kötter and Kschischang proposed an approach that finds a linear subspace of the ambient vector space, and the coding is just done in that linear subspace [KK08]. This is a challenging task since every combination of source data should result in a distinct union subspace and finding a proper subspace can be a computational challenge.

The concept of sparse coding is well known, and it was first applied for header compression in network coding by Siavoshani et al. [SKFA09]. The number of combined packets in one coded packet is reduced from n to m, where $m < n$, which results in a header length of $\mathcal{O}(m \log_2 n \log_2 q)$ bits. However, limiting the number of combined packets affects the invertibility of the matrix or it reduces the probability of a redundant packet being innovative [BKW97, HPFM11, FLS$^+$14, PFS05]. It was proved that m should be at least of order $\mathcal{O}(\log n)$ so that the matrix has a full rank with high probability.

A header compression algorithm based on erasure decoding and list decoding was presented in [LR10]. The compressed header length under the erasure decoding scheme is $m + n/\log_2 q$ bits. The header length becomes arbitrarily close to $m + O(\log_2 n)/\log_2 q$ bits when the list decoding scheme is used. Both schemes are valid for moderate or large values of m.

In [CCW10], the header overhead is the seed for generating the coding coefficients with a known Pseudo-Random Number Generator (PRNG). This effectively reduces the header overhead to the size of the seed, but it does not support re-encoding which is the crucial constituent of RLNC [LWLZ10].

A similar solution where the generation of the coding coefficients is based on modified Vandermonde matrices which can be determined by one symbol is given in [TF12]. Two main drawbacks of this solution are: the network coding nodes can only perform addition operations and the generation size is upper bounded by $\log_2 q$ bits due to the cyclic property of the matrices.

Silva showed that precoding with Maximum Rank Distance (MRD) codes virtually eliminates the linear dependency even over a binary field [Sil12]. Coding in small finite fields significantly reduces the overhead in RLNC. This approach implies a moderate increase in the decoding complexity, but it potentially simplifies the operations at intermediate nodes that comes as an additional benefit besides minimizing the total overhead.

Recently, Fulcrum codes were proposed [LPHF14b, LPHF14a]. Fulcrum codes are concatenated codes where a seed for a PRNG is used to end-to-end communicate the coefficients of the outer code, while the inner code requires $1+r/n$ bits per packet. Recoding can be performed exclusively over the inner code in $GF(2)$. Encoding and decoding is performed over the outer code in big finite fields.

Although there has been a vast amount of research results for network coding since its emergence, still there have not been many practical applications. The concept

of network coding has been used to derive the bounds of the repair bandwidth in distributed storage systems. This is discussed in the next subsection.

The research in the present thesis addresses one of the main limitations for practical implementation of network coding:

– Reducing the length of the vector of coding coefficients.

2.3 Code Constructions for Distributed Storage Systems

A *distributed storage system* is a network of storage disks or nodes where data pertaining to a single file is distributed across the storage nodes. It is a practical choice for storing large amounts of data. The nodes are relatively inexpensive storage devices that may fail, be down during maintenance, or otherwise unavailable due to serving other demands, etc. A distributed storage system has to guarantee a reliable storage of the data over long periods of time even though the nodes might be individually unreliable.

Definition 2.17. *Reliability* is the probability that a system provides an uninterrupted service during a certain time interval $[0, t]$, i.e.

$$R(t) = P(T_{FF} > t), \tag{2.9}$$

where T_{FF} is Time to First Failure.

Definition 2.18. Mean Time to First Failure (MTFF) or Mean Time To Data Loss (MTTDL) is a measure of the reliability of a system defined as

$$MTFF = \int_0^\infty R(t)dt. \tag{2.10}$$

It equals the time it takes a given storage system to exhibit enough failures such that at least one block of data cannot be retrieved or reconstructed.

In order to build a highly reliable system, redundancy has to be introduced. The redundancy can either be a simple copy of the data or a linear combination of the original data. *Replication* is a method of making copies from the original data. The data is available until one copy still exists. In case of storing one extra replica, the storage overhead is 100%, while it is 200% for 2 replicas and so forth. For instance, Google File System [GGL03] and Hadoop File System [SKRC10] store three copies of the data by default. When storing petabytes of data, replication is cost-inefficient. The main advantages of replication are: simple design and verification, low I/O and latency. However, its major disadvantage is the high storage overhead (200% for the industry standard) that translates into a high hardware cost (disk drives and associated equipment), as well as a high operational cost such as building space, power, cooling, maintenance, etc.

Weatherspoon and Kubiatowicz showed in [WK02] that erasure resilient systems use an order of magnitude less bandwidth and storage to provide a similar level of reliability as replicated systems. Hence many distributed storage systems are now turning to Reed-Solomon codes. Two reasons why often RS codes are employed in large-scale distributed systems are their storage optimality (since they are MDS codes) and generic applicability (construction of RS codes for arbitrary n and k). Redundant Arrays of Inexpensive Disks (RAID) is a well known technology for data protection in high performance computing storage systems [PGK88]. RAID based systems use RS codes to recover the data when multiple disks fail simultaneously. In order to avoid the large finite field operations, many code constructions such as EvenOdd [BBBM95], Row-Diagonal Parity [CEG$^+$04] and flat XOR [GLW10] use only exclusive-OR operations to recover the data. Although RS codes improve the storage efficiency, the amount of accessed and transferred data to repair a failed node is large. We graphically present the repair process of a single systematic node with RS code in Figure 2.10 (a). The parameters in this example are $k = 10$, $r = 4$ and $M = 100$MB. When RS codes are used, each node is recovered by transferring data of size M/k from any k nodes. In order to reconstruct 10MB of data stored in a single node, 10×10MB=100MB are read from 10 nodes and transferred across the network to the node performing the decoding computations. Accordingly, RS codes perform poorly in terms of the repair bandwidth since the amount of the repair bandwidth is k times the size of the data to be reconstructed. Additionally, this has a negative effect on the read performance of the system in a degraded mode (there is a read request for a data unit that is missing or unavailable) and the recovery time.

An improvement is to seek for codes that perform better than RS codes. Three major repair cost metrics for new erasure coding solutions have been identified in the recent literature: i) the amount of transferred data during a repair process (*repair bandwidth*), ii) the number of access operations during each repair (*disk-I/O*), and iii) the number of nodes that participate in the repair process (*repair locality*). Two types of repairs have been suggested[DRWS11]: *exact* (the recovered data has exactly the same content as the lost data) and *functional* (the newly generated data can be different than the lost one, but it maintains the MDS-property). The research presented in the present thesis focuses only on exact repair, which is preferred from a practical point of view.

2.3.1 Regenerating Codes

Classical MDS codes are optimal in terms of the storage-reliability tradeoff, but they still do not give a good answer to the key question: *How to encode the data in a distributed way while transferring as little data as possible across the network during a repair process?*

Under a (n, k) MDS code, a file of size M symbols is divided into k fragments, each of size $\frac{M}{k}$ symbols, encoded and stored in n nodes. The original file can be

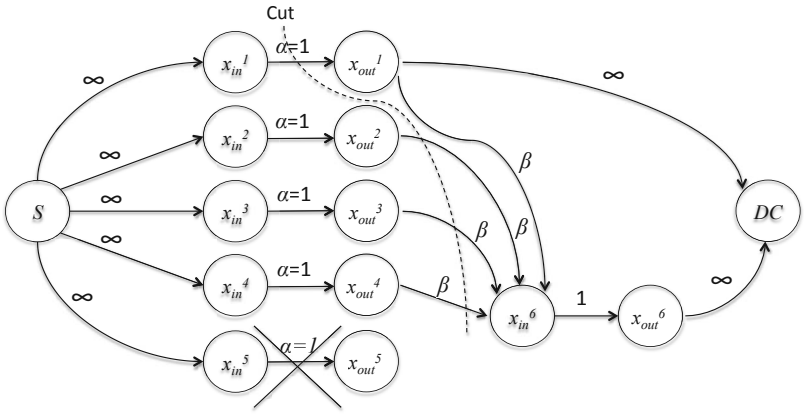


Figure 2.8: Illustration of an information flow graph corresponding to a $(5, 3)$ MDS code. When node x_5 is unavailable, a new node x_6 reconstructs the data by connecting to $d = 4$ available nodes and downloading βMB from each node.

recovered from any set of k fragments. Hence any data collector would preferably connect to these k nodes. Dimakis et al. [DGW$^+$10] showed that the repair problem under functional repair can be mapped to a multicasting problem in an information flow graph [DGW$^+$10]. Building on known results from network coding, they proved that a node repair is possible if and only if the underlying information flow graph has sufficiently large min-cuts. Figure 2.8 shows an information graph that corresponds to a $(5, 3)$ code where the node S corresponds to the source file. Assume that a $(5, 3)$ MDS code is used to encode the file $M = 3$MB. The code generates 5 fragments each of size $\alpha = 1$MB (stored in the nodes $x_1, \ldots, x_5$) with the property that any 3 can be used to reconstruct the original data. The storage node x_i is represented by a storage input node x_{in}^i and a storage output node x_{out}^i. The capacity of the edge between these two nodes is equal to the amount of stored data in node x_i. Data collectors connect to subsets of active nodes through edges with infinite capacity. When node x_5 fails, a newcomer x_6 reconstructs the lost data from $x_1, \ldots, x_4$. The question is: What is the minimum amount of information that has to be communicated? The new storage node x_6 downloads βMB from $d = 4$ active nodes. The min-cut separating the source and the data collector must be larger than $M = 3$MB for a reconstruction to be possible. In this example, the min-cut value is given by $1 + 3\beta$, implying that $\beta \geq 0.33$MB is sufficient and necessary.

A (n, k, d) regenerating code replaces the data from a failed node by downloading β symbols from d non-failed nodes where $\beta \leq \alpha$. Thus, the repair bandwidth is $\gamma = d\beta$ where $\alpha \leq \gamma \ll M$. The main result of [DGW$^+$10] is the following Theorem.

Theorem 2.19. *[DGW$^+$10, Th.1] For any $\alpha \geq \alpha^*(n, k, d, \gamma)$, the points $(n, k, d, \alpha, \gamma)$ are feasible, and linear network codes suffice to achieve them. It is information theo-*

retically impossible to achieve points with $\alpha < \alpha^*(n,k,d,\gamma)$*. The threshold function* $\alpha^*(n,k,d,\gamma)$ *is the following:*

$$\alpha^*(n,k,d,\gamma) = \begin{cases} \frac{M}{k}, \gamma \in [f(0), +\infty) \\ \frac{M-g(i)\gamma}{k-i}, \gamma \in [f(i), f(i-1)) \end{cases} \tag{2.11}$$

where

$$f(i) \triangleq \frac{2Md}{(2k-i-1)i + 2k(d-k+1)} \tag{2.12}$$

$$g(i) \triangleq \frac{(2d-2k+i+1)i}{2d} \tag{2.13}$$

where $d \leq n-1$*. For* d, n, k *given, the minimum repair bandwidth* γ *is*

$$\gamma_{min} = f(k-1) = \frac{2Md}{2kd-k^2+k}. \tag{2.14}$$

The repair bandwidth γ decreases as the number of d nodes increases. An optimal tradeoff curve between the storage and the repair bandwidth for a $(15, 10, 14)$ code is shown in Figure 2.9. Regenerating codes achieve each point on this optimal tradeoff curve. The two extremal points on the optimal tradeoff correspond to the points at which the storage and the repair bandwidth are minimized, respectively. Following Theorem 2.19, the both points are attained when $d = n-1$, i.e. all $n-1$ non-failed nodes called helpers are contacted during a node repair. The codes that attain these points are known as *Minimum-Storage Regenerating (MSR)* and *Minimum-Bandwidth Regenerating (MBR)* codes, respectively. MSR codes achieve the pair

$$\left(\alpha_{MSR}, \gamma_{MSR}^{min}\right) = \left(\frac{M}{k}, \frac{M}{k}\frac{n-1}{n-k}\right), \tag{2.15}$$

while MBR codes

$$(\alpha_{MBR}^{min}, \gamma_{MBR}^{min}) = \left(\frac{M}{k}\frac{2n-2}{2n-k-1}, \frac{M}{k}\frac{2n-2.}{2n-k-1}\right). \tag{2.16}$$

MBR codes require an expansion factor of $\frac{2n-2}{2n-k-1}$ in the amount of stored data, thus, they are no longer storage-reliability optimal.

The work in [CJM10] and [CR10] showed that the lower bound of the repair bandwidth for functional repair with MSR codes given in (2.15) is also achieved for exact repair with MSR codes.

The present thesis focuses only on storage-reliability optimal codes with exact repair since they have the biggest practical potential.

We illustrate the benefit of using an MSR code in Figure 2.10 (b). We use the same parameters as for the RS code, i.e. $k = 10$, $r = 4$ and $M = 100$MB. Under the MSR code, the data from an unavailable node is reconstructed by downloading only $\frac{100}{40}$MB=2.5MB from each of the 13 non-failed nodes, i.e. $\frac{1}{4}$ of the stored data from

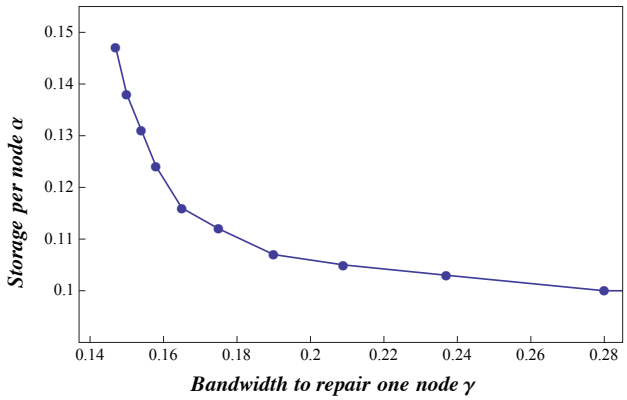


Figure 2.9: An optimal tradeoff curve between the storage α and the repair bandwidth γ for a $(15, 10, 14)$ code and $M = 1$ [DGW$^+$10]. Traditional erasure coding (RS codes) corresponds to the points $\alpha = 0.1$ and $\gamma = 1$.

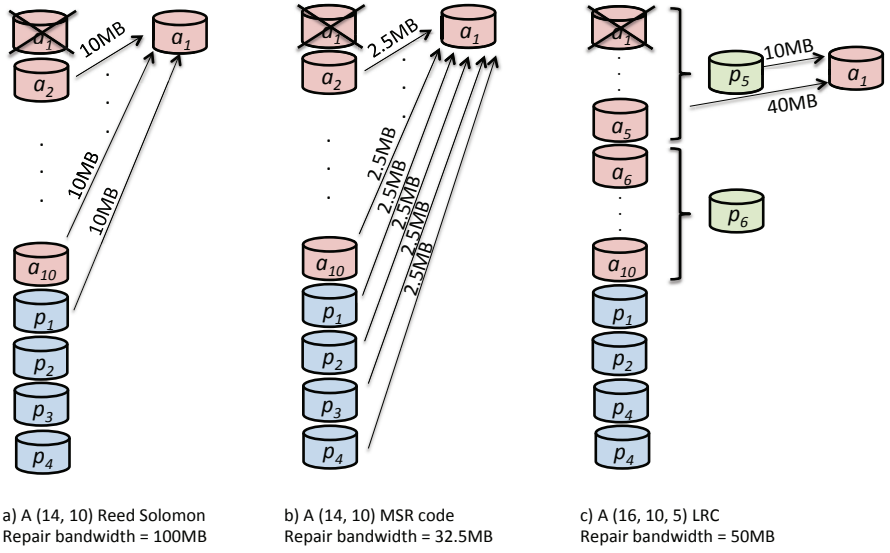


Figure 2.10: Amount of transferred data for reconstruction of the systematic node a_1 for a (14,10) RS code, a (14, 10) MSR code and a (16, 10, 5) LRC. The systematic nodes are represented in red and the parity nodes in blue, while the local parity nodes for the LRC are in green.

each node. The total repair bandwidth is only 32.5MB compared to 100MB under RS.

MSR codes possess all properties of an MDS code, giving an additional advantage of efficient repair consuming minimum possible bandwidth. This is made possible by using a sub-packetization level, i.e. the data at each node is further divided into blocks. The sub-packetization level α represents the minimum dimension over which all operations are performed. Namely, when the sub-packetization is 1, then each node is recovered by transferring data of size M/k symbols from any k nodes, i.e. the case when RS codes are used. Hence, $\alpha > 1$ is required to achieve the lower bound of the repair bandwidth. We use a sub-packetization level equal to 64 in Figure 2.10 (b) in order to achieve the minimum repair bandwidth with the $(14, 10)$ MSR code.

The construction of exact-repair codes is a well-studied problem in the literature. Here we only list the most relevant results for our work about MDS codes for storage systems. Exact-MSR codes that are obtained by using the technique of interference alignment, a technique used to efficiently handle multiple interfering channels in wireless communications [CJ08], were presented in [WD09, CR10].

Papailiopoulos et al. in [PDC13] and Cadambe et al. in [CHL11] resolved partly the open problem about designing high-rate MDS codes that achieve the optimal repair bandwidth. The code construction in [PDC13] used Hadamard matrices to construct a two-parity MDS code with optimal repair properties for any single node failure, including the parities. The construction is similar to zigzag codes in [TWB13], but the former one uses bigger finite fields. The zigzag codes provide an optimal recovery of any systematic node and an optimal update for a sub-packetization level of r^k. The work was further extended to provide an optimal recovery of both systematic and parity nodes [WTB11].

Furthermore, Tamo et al. [TWB14] showed that the sub-packetization level of an access-optimal MDS code for repairing a failed systematic node is $r^{\frac{k}{r}}$. In [CHLM11], Cadambe et al. proposed codes that repair optimally any systematic node for this sub-packetization level. An alternate construction of access-optimal MSR codes motivated by zigzag codes was presented in [ASVK15]. An essential condition for designing alternate codes with r parity nodes is $m = \frac{k}{r}$ to be an integer $m \geq 1$ where k is set to rm and α to r^m.

Wang et al. constructed codes that optimally repair any systematic or parity node and require a sub-packetization level of r^{k+1} [WTB11]. High-rate MSR codes with polynomial sub-packetization level were proposed in [SAK15]. However, the work presented in the present thesis focuses only on optimal repair of any systematic node.

Although the aforementioned MSR constructions achieve the lower bound of the repair bandwidth for a single failure, they have not been practically implemented in real-world distributed storage systems. Two main reasons for practical abandonment of existing MSR codes are: either MSR codes require encoding/decoding operations

over an exponentially growing finite field or the sub-packetization level α increases exponentially. There are at least two ways to solve the problem by using small sub-packetization levels and optimizing in terms of I/O operations.

Rashmi et al. reported 35% reduction in the repair bandwidth for any systematic node when the sub-packetization level is 2 compared to a $(14, 10)$ RS code [RSG$^+$14]. They used the piggyback framework [RSR13] to construct Hitchhiker erasure codes. The basic idea of the piggyback framework is to take multiple instances of an existing code and add carefully designed functions of the data from one instance to another. Other code constructions with small sub-packetization levels are Rotated-RS [KBP$^+$12], EVENODD [BBBM95] and RDP codes [CEG$^+$04]. Rotated-RS codes exist only for $r \in \{2, 3\}$ and $k \leq 36$, while EVENODD and RDP codes exist for $r = 2$.

I/O is becoming the primary bottleneck in the performance of cloud storage systems and applications that serve a large number of user requests or perform data intensive computations such as analytics. There are two main types of I/Os: sequential and random operations. *Sequential operations* access locations on the storage device in a contiguous manner, while *random operations* access locations on the storage device in a non-contiguous manner. A recent algorithm to transform Product-Matrix-MSR codes [RSK11] into I/O optimal codes (termed PM-RBT codes) while retaining their storage and network optimality was presented in [RNW$^+$15]. PM-RBT exist only for $r \geq k - 1$, i.e. for the low-rate regime.

All MDS erasure codes discussed in the previous paragraphs focus on an efficient repair from a single failure, since single failures present 98.08% of the total failures [RSG$^+$14]. On the other hand, the authors in [FLP$^+$10] state that the failures are often correlated. Next we review codes that outperform the aforementioned codes when multiple failures happen.

A cooperative recovery mechanism in the minimum-storage regime for repairing from multiple failures was proposed in [HXW$^+$10, WXHO10]. Minimum Storage Collaborative Regenerating (MSCR) codes minimize the repair bandwidth while still keeping the MDS property by allowing new nodes to download data from the non-failed nodes and the new nodes to exchange data among them. The repair bandwidth for MSCR codes under functional repair was derived independently in [HXW$^+$10, KSS11]. The existence of a random linear strong-MDS code under the assumption that the operations are in a sufficiently large finite field was showed in [HXW$^+$10]. The codes attain the MSR point but the decoding complexity is quite expensive. Adaptive regenerating codes where the numbers of failed and surviving nodes change over time were proposed in [KSS11]. The authors in [LL14] showed that it is possible to construct exact MSCR codes for optimal repair of two failures directly from existing exact MSR codes. MSCR codes that cooperatively repair any number of systematic nodes and parity nodes or a combination of one systematic and one parity node were presented in [CS13]. However, the code rate of these codes

is low ($n = 2k$). A study about the practical aspects of codes with the same code rate ($n = 2k$) in a system called CORE that supports multiple node failures can be found in [LLL15]. There is no explicit construction of high-rate MDS codes for exact repair of multiple failures at the time of writing of the present thesis.

Practical scenarios [SAP+13, PJBM+16, RSG+14] showed that erasure codes have to provide a satisfactory tradeoff between the metrics such as storage overhead, reliability, repair bandwidth, locality and I/Os. The present thesis tackles this by working on:

- MDS code constructions with arbitrary sub-packetization levels for both low-rate and high-rate regimes; and
- Locally Repairable Codes.

2.3.2 Locally Repairable Codes

Locally Repairable Codes (LRCs) were independently introduced in [GHSY12, OD11, PLD+12].

Let C be a $(n, k, d)_q$ linear code. Assume that the encoding of $\mathbf{x} \in \mathbf{F}_q^k$ is by the vector

$$C(\mathbf{x}) = (\mathbf{c}_1 \cdot \mathbf{x}, \ldots, \mathbf{c}_n \cdot \mathbf{x}) \in \mathbf{F}_q^n. \tag{2.17}$$

Thus, the code C is specified by the set of n points $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_n\} \in \mathbf{F}_q^k$. The set of points must have a full rank equal to k for C to have k information symbols.

Definition 2.20. For $\mathbf{c}_i \in C$, we define $Loc(\mathbf{c}_i)$ to be the smallest integer l for which there exists $L \subseteq C$ of cardinality l such that

$$\mathbf{c}_i = \sum_{j \in L} \lambda_j \mathbf{c}_j. \tag{2.18}$$

We further define $Loc(C) = max_{i \in \{1,\ldots,n\}} Loc(\mathbf{c}_i)$.

Definition 2.21. We say that a code C has information locality l if there exists $I \subseteq C$ of full rank such that $Loc(\mathbf{c}) \leq l$ for all $\mathbf{c} \in I$.

Gopalan et al. derived the upper bound for the minimum distance of a $(n, k, d)_q$ code with locality l.

Theorem 2.22. *[GHSY12, Th.5] For any $(n, k, d)_q$ linear code with information locality l,*

$$d \leq n - k - \left\lceil \frac{k}{l} \right\rceil + 2. \tag{2.19}$$

Huang et al. showed the existence of Pyramid codes that achieve the distance given in (2.19) when the field size is big enough [HCL13].

Two practical LRCs have been implemented in Windows Azure Storage [HSX+12] and HDFS-Xorbas by Facebook [SAP+13]. Both implementations reduce the reconstruction cost by introducing l local and r global parity blocks. The local parity nodes are computed from a subset of the systematic nodes. The locality of a code has also an impact on the fault tolerance and the update efficiency. LRCs tolerate $r+1$ *arbitrary node failures* and $l+r$ *theoretically decodable failures*. Consider the example with 10 data nodes in Figure 2.10 (c) where a (16, 10) LRC generates 6 (instead of 4) parity nodes. The first four parities (denoted as p_1, p_2, p_3, p_4) are global parities and are computed from all systematic nodes. While, for the two other parities, LRC divides the systematic nodes into two equal size groups and computes one local parity node for each group. The local parity p_5 is computed as an XOR combination from 5 systematic nodes in the first group $(a_1, \ldots, a_5)$, while the parity p_6 is computed from 5 systematic nodes in the second group $(a_6, \ldots, a_{10})$.

Let us consider the reconstruction of a_1. Instead of reading p_1 (or another global parity node) and the remaining 9 systematic nodes, it is more efficient to read p_5 and 4 systematic nodes $(a_2, \ldots, a_5)$ from the first group. It is easy to verify that the reconstruction of any systematic node requires accessing only 5 nodes, i.e. significantly less than the RS (10 nodes) and the MSR (13 nodes) code. The locality in this example is equal to 5 for the local parities and equal to 10 for the global parities. Any systematic node is recovered from $\frac{k}{l}$ nodes within its local group. Hence, the repair bandwidth for a single systematic node recovery with the (16, 10) LRC is 50MB. If we consider a repair of the parity nodes as well, then the average repair bandwidth for a single node failure is $(5 \times 12 + 10 \times 4) \times 10\text{MB}/16 = 62.5\text{MB}$, because the recovery of the global parities is performed in the same way as with RS codes.

The authors of [SAP+13] improved the recovery of the global parities by introducing an implied parity, but choosing the coefficients for the parities to satisfy the alignment condition is computationally demanding. Tamo et al. introduced a new family of optimal LRCs that are based on re-encoding RS encoded fragments [TPD13]. Although the code construction is simple, it requires a large finite field.

Motivated by practical situations in hot storage, where the data changes dynamically, the metric *update complexity* has been introduced in [ASV10]. Specifically, if the value of any of the systematic data changes, then the corresponding data has to be updated in the nodes that contain it in order to keep the data consistent.

Definition 2.23. The *update complexity* of a $(n, k, d)_q$ code C is defined as the maximum number of symbols that must be updated when any single information symbol is changed.

Tolerating and recovering efficiently from multiple failures is an important requirement for big data storage systems. Shingled erasure codes (SHEC) are codes with local parities shingled with each other that provide efficient recovery from multiple failures [MNS14]. All parities have the same locality l and support $\frac{rl}{k}$ systematic or

parity node failures without data loss, but they are not efficient in terms of storage overhead and reliability.

There is still a need for new erasure resilient codes that reduce the number of nodes contacted during a repair, while still guaranteeing a low repair bandwidth even when multiple failures occur. Additionally, performing updates consumes bandwidth and energy, so it is of a great interest to construct codes that have small update complexity, i.e. small locality. Thus, one of the research topics in the present thesis is:

– A construction of LRCs that have low storage overhead, low average repair bandwidth for single and double failures, high reliability and improved update performance.

2.4 Optical Packet Switched Networks

Dense Wavelength Division Multiplexing (DWDM) has emerged as a core transmission technology for backbone networks. With DWDM, optical signals are multiplexed enabling simultaneously different wavelengths on the same fiber. Fiber networks can therefore carry multiple Terabits of data per second over thousands of kilometers [htt]. ADVA optical networking reported that the current DWDM systems support up to 192 wavelengths on a single fiber, with each wavelength transporting up to 100Gbit/s – 400Gbit/s and 1 Terabit/s.

However, DWDM opaque networks use expensive optical/electrical/optical (O/E/O) conversion for switching. Although the speed of electronic devices has been increased significantly, it is still not likely to catch up with the transmission speed available at the optical layer. Thus, there is a need to minimize or eliminate the electronic processing in order to fully exploit the potential bandwidth offered by DWDM. This calls for a move of the switching functionalities from the electronic domain to the optical domain, i.e. all-optical networking [OSHT01].

Furthermore, the traffic has become more data-dominant than voice-dominant. As the traffic nature has changed from continuous to bursty, there is a need for a switching technology that supports efficiently bursty traffic [QY99].

The switching technologies for DWDM are:

– *Wavelength Routed Optical Switching* establishes all-optical circuit switched connections (lightpaths) between edge nodes in the optical core network [RS02]. In Wavelength Routed Optical Networks (WRON), a lightpath is set-up before the data transmission and a dedicated wavelength on every link is reserved. The lightpath may be wavelength converted at the intermediate nodes. Thus, WRON does not require buffering, O/E/O conversion or processing at intermediate nodes. The major problem in WRON is the non-optimal utilization of link resources, because there is no resource sharing between lightpaths traversing the same link.

Table 2.1: A comparison of switching technologies for DWDM [VCR00].

Switching technology	Granularity	Utilization	Complexity
Wavelength Switching (WRON)	Coarse	Poor	Low
Optical Packet Switching (OPS)	Fine	High	High (not mature)
Optical Burst Switching (OBS)	Moderate	Moderate	Moderate

– An alternative to Wavelength Routed Optical Switching is *Optical Packet Switching (OPS)* and *Optical Burst Switching (OBS)* [GRG+98, YSH+98] that enable all-optical packet transport combined with statistical multiplexing for increased link utilization [HA00, CQY04, Tur99].

One of the main differences between OPS and OBS is the data unit that is processed and forwarded through the network. In OPS networks, packets are processed and forwarded, while in OBS networks, incoming packets are aggregated into bursts at an OBS ingress node based on the destination and/or the service class of the packets which are then transmitted in the network. Both OPS and OBS allow switching of data in the optical domain. However, switching decisions are made in the electronic domain [NBG08] as optical processing is still an immature technology.

Another key difference between OPS and OBS is the control information. The control information in OPS is in-band, while it is out-of-band in OBS networks. In particular, reservation is not possible in OPS, because the header follows the rest of the packet. In OBS, the burst transmission is initiated shortly after the burst was assembled and the control packet was sent out. The wavelength is allocated only for the duration of a data packet/burst and can be statistically shared by packets/bursts belonging to different connections. Switching decisions are taken based on the packet header or the burst control packet that undergoes O/E conversion and electronic processing at the switches, while the packet/burst payload is optically switched.

OPS has finer granularity compared to OBS, but requires fast switching, header reading and reinsertion that increase the complexity and the cost at the switching node [NBG08].

An overview of the characteristics of the aforementioned switching technologies is given in Table 2.1 [VCR00].

A crucial issue in OPS/OBS networks is the *packet loss at the network layer* due to contentions. A *contention* occurs at a switching node when two or more packets are destined on the same output port, on the same wavelength, at the same time. Contending packets are dropped which leads to increased PLR. For instance, the default behavior is to drop one of the two packets that contend for the same wavelength at the same time. In Figure 2.11 (a), packet *b* is dropped. It is impossible

to buffer the data or retransmit the dropped packets. First, the amount of data is enormous, and, second, there is no Random-Access Memory (RAM) available in OPS/OBS networks. Thus, contending packets cannot be buffered and forwarded when the output port is free.

Several contention resolution mechanisms such as wavelength conversion, Fiber Delay Line (FDL) buffering and deflection routing [CWXQ03] have been proposed to combat packet loss in OPS/OBS networks. Wavelength conversion [DJMS98, EM00] converts the wavelength of one of the contending packets to an idle wavelength on the same output port (Figure 2.11 (b)). FDLs [HCA98] try to mimic electronic buffering by delaying one of the packets in time and scheduling it to the intended wavelength when it is free (Figure 2.11 (c)). A large number of FDLs are needed to implement large buffer capacity and they may add an additional delay. Both wavelength conversion and optical buffering require extra hardware which may increase the system cost. Deflection routing is a multiple-path routing technique that routes the contending packets to other nodes, i.e. output ports on the same wavelength. The performance of deflection routing largely depends on the network topology. A big advantage is that any of these three techniques can be combined. Another contention resolution scheme in OBS networks is burst segmentation. Rather than dropping the entire burst during contention, only the overlapping segments are dropped [VJS02].

FEC has been recently applied in OPS/OBS networks to alleviate PLR (Figure 2.11 (d)) [YKSC01]. FEC is not a contention resolution scheme, which means it can be combined with wavelength conversion, fiber delay line buffering and deflection routing. Redundant packets are added to a set of data packets at the OPS ingress node and transmitted along with the original data packets to an OPS egress node. Data packets dropped due to a contention can be reconstructed at the OPS egress node by using excess redundant packets, leading to a potential reduced PLR. The Network Layer Packet Redundancy Scheme (NLPRS), introduced in [Øve04], reduces several orders of magnitude the end-to-end PLR due to contentions in an asynchronous OPS ring network with and without a wavelength conversion. The authors in [GA02] evaluate several topology-routing algorithms for deflection routing coupled with erasure coding. The results show the amount of redundancy that is needed in unstructured networks of switches. The forward redundancy mechanism proposed in [VZ06] significantly reduces the packet loss compared to a retransmission-based backward loss recovery mechanism without the need for large ingress electronic buffers or big retransmission delays. Under this mechanism, only the overlapping segments of the contenting bursts are dropped. The dropped segments of a burst can be recovered using the redundant packets at the OBS egress node that were sent in the forward direction, from the ingress to the egress node.

Apostolopoulos used multiple state video coding and path diversity for transferring video over a lossy packet network [Apo01]. *Path diversity* is defined as sending different subsets of packets over disjoint paths, as opposed to the default scenarios

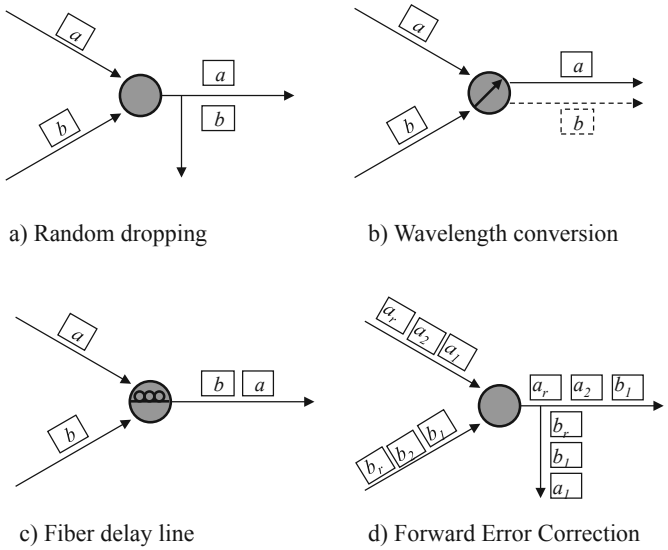


Figure 2.11: Contention resolution mechanisms at an OPS node where the packets a and b arrive on the same wavelength at the same time and contend for the same output wavelength. a) The packet a is transmitted, while b is dropped; b) Contention resolution with wavelength conversion where packet b is converted to an idle wavelength (on the same fiber); c) Contention resolution with FDL buffering where packet b is delayed using FDL buffering; d) Contention resolution with FEC where redundant packets are added.

where the packets proceed along a single path. Path diversity provides better performance because the probability that all of the multiple paths are simultaneously congested is much less than the probability that a single path is congested. Nguyen and Zakhor combined a rate allocation algorithm from multiple senders to a receiver with FEC in order to minimize the probability of lost packets due to congestions in a bursty loss environment [NZ02]. The work was further extended in [NZ03] where they presented a scalable, heuristic scheme for selecting a redundant path between an ingress node and an egress node. The disjoint paths from a sender to a receiver are created by using a collection of relay nodes.

Another coding technique for mitigating packet loss is network coding. A straightforward application of the network coding technique is not feasible in OPS/OBS due to the lack of store-and-forward capabilities. However, the idea of combining the packets instead of dropping them reduces the packet loss [BØ11].

Survivability is essential in OPS/OBS networks with a throughput of order of terabits per second [ZS00]. The *survivability* of a network refers to a network's capability to provide continuous service in the presence of failures. The most common types of failures are node and link failure. Service providers have to

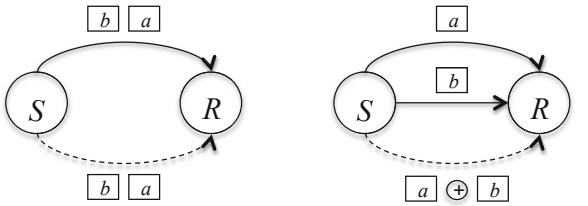


Figure 2.12: Different path protection schemes.

ensure that their networks are fault tolerant. To meet these requirements, providers use common survivability mechanisms such as predesigned protection techniques [GL03, FV00, SRM02], node and component redundancy, prebuffering [KBØS10], and predetection schemes [BNH02]. The most used protection techniques include $1+1$ protection, in which the same data is transmitted on two link disjoint paths, and the receiver selects the packets from the path with better quality; $1:1$ protection, which is similar to $1+1$, except that traffic is not transmitted on the backup path until a failure occurs; $1:N$ protection, which is similar to $1:1$, except that one path is used to protect N paths; and $M:N$, where M protection paths are used to protect N working paths. Figure 2.12 shows $1+1$ and $1:2$ path protection schemes.

Some of the protection techniques can be combined with coding. Kamal first applied network coding to provide $1+N$ protection against single link failure [Kam06, Kam07b, Kam08]. The optimal $1+N$ protection scheme in [KAK08] requires exactly the same amount of protection resources as in $1:N$ and the time to recover from failures is comparable to that of $1+1$ protection. Kamal extended the approach to protect against multiple link failures [Kam07a, KRLL11]. Menendez and Gannet proposed photonic XOR devices for network coding [MG08]. Savings of up to 33% in links (transmitter, fiber or wavelength channel and receiver) are possible with network coding compared to conventional techniques. The effectivness of network coding to provide robustness against link failures of multicast traffic is presented in [MDXA10].

In OBS, data and control packets are sent out of band. Sending the Burst Header Packet (BHP) prior the transmission of data packets with specific offset time exposes the data payload to different security challenges. The authors in [SMS12] discussed the burst hijacking attack where a source node can maliciously create a copy of the original BCH and modify its value to setup a path to a malicious destination. The data payload is forwarded to the original destination as well as the malicious destination. However, the malicious destination does not send an acknowledgment for this hijacked burst, thus, it escapes from being caught. A solution based on Rivest-Shamir-Adleman (RSA) public-key encryption algorithm

that addresses the Data Burst Redirection (DBR) attack in OBS networks is proposed in [CAKSAL+15]. However, the high capacities of OPS/OBS networks make data encryption in OPS/OBS not feasible as the current computational resources do not match the required encryption processing demands. The encryption mechanisms have to be with low computational complexity, suitable for high-speed implementation and the majority of the header content should not be encrypted since the processing of the headers has to be at ultra high speed [CV08].

One way of providing a certain level of security in a non-cryptographic way is to utilize non-systematic coding. *Secrecy* as defined in [MS12] provides protection from a passive adversary that is not able to reconstruct the whole packet/burst set by eavesdropping on a single path. This property has not been so far exploited in OPS/OBS networks. The authors in [OLV+12] showed how secrecy in storage systems is provided even when an eavesdropper knows or can guess some of the missing information.

There is a lack of research work that gives a unified view of the performance metrics in OPS. The scheme proposed in [Øve08] and [ØBBT12] provides a $1 + 1$ path protection in addition to the packet loss alleviation. In particular, the work in [ØBBT12] shows that significant cost savings are achieved by using erasure codes compared to other approaches that provide 1+1 path protection. There is a need for:

– An integrated view of QoS in OPS/OBS networks that deals with survivability, packet loss alleviation and secrecy at the same time.

Chapter 3
Contributions and Concluding Remarks

3.1 Research Questions

The state-of-the-art in the four main research areas covered in the present thesis was reviewed in the previous Chapter. The detailed literature study brings forth the following research questions.

First, it is of utmost importance to identify the desired code properties explained in Subsection 2.1 when constructing erasure codes, properties such as non-MDS or MDS, binary or non-binary codes with as few as possible operations in large finite fields, structure of the generator matrix, generic applicability, fast encoding/decoding etc. Accordingly, it is meaningful to ask:

RQ1 How can we construct balanced erasure codes?

The next goal is to address some of the challenges of a practical implementation of network coding. Network coding can increase the data throughput by an order-of-magnitude and also improve the robustness of existing networks. However, one of the main challenges is the header overhead imposed by the coding coefficients as explained in Subsection 2.2. Accordingly, there is an enormous need for new header compression algorithms and this poses the next research question:

RQ2 How can we reduce the header overhead in network coding?

Network coding concepts helped to derive the storage-repair bandwidth tradeoff for single node recovery with regenerating codes in distributed storage systems. While MSR codes possess all properties of MDS codes and offer an additional advantage of efficient repair consuming the minimum repair bandwidth (Subsection 2.3.1), locally repairable codes disregard the MDS property and provide a low locality (Subsection 2.3.2). Thus, a more general question about code constructions that are optimal for some of the cost metrics in distributed storage systems is raised:

RQ3 How can we construct efficient codes for distributed storage systems?

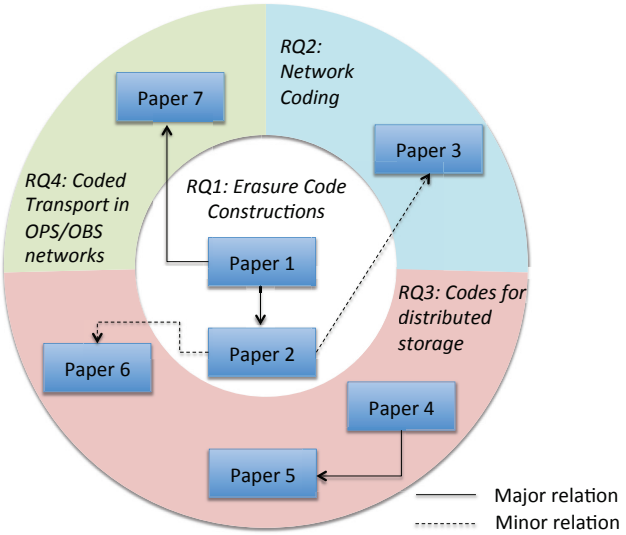


Figure 3.1: Relations between the papers included in the thesis. The papers are grouped based on the research questions.

The next question is related to provision of a unified view on QoS in OPS/OBS networks that is explained in Subsection 2.4. It is essential to know the interactions between the survivability, the packet loss rate and the secrecy when erasure codes are applied. Naturally, this poses the following question:

RQ4 How can erasure coding be applied to increase the QoS in OPS/OBS networks?

3.2 Research Results

The author of the present thesis wrote and contributed to 10 publications and 7 patent applications during the four-year PhD period. Table 3.1 presents a complete list of included publications in the thesis, while Table 3.2 and Table 3.3 list the rest of the publications and patent applications that are not included in the thesis. The order in which the papers are given is not necessarily chronological, but rather related to the research questions so that it is easier and more natural to follow the exposition in the thesis.

Figure 3.1 shows the position of the included papers in the thesis with regards to the four research questions. Erasure code constructions are the core and they are used in different applications. The arrows depict the correlations between the included papers. Combinatorics and coding theory go hand in hand acting as powerful tools for design, analysis and implementation of efficient codes. We started with construction of erasure codes from combinatorial designs where we generate codes

Table 3.1: List of publications included in the thesis.

Paper	Title · Author List · Conference/Journal
Paper 1	**Balanced XOR-ed Coding** K. Kralevska, D. Gligoroski, and H. Øverby Lecture Notes in Computer Science, vol. 8115, pp. 161-172, 2013
Paper 2	**Families of Optimal Binary Non-MDS Erasure Codes** D. Gligoroski and K. Kralevska IEEE Proceedings on International Symposium on Information Theory (ISIT), pp. 3150-3154, 2014
Paper 3	**Minimal Header Overhead for Random Linear Network Coding** D. Gligoroski, K. Kralevska, and H. Øverby IEEE International Conference on Communication Workshop (ICCW), pp. 680 - 685, 2015
Paper 4	**General Sub-packetized Access-Optimal Regenerating Codes** K. Kralevska, D. Gligoroski, and H. Øverby IEEE Communications Letters, vol. 20, issue 7, pp. 1281 - 1284, 2016
Paper 5	**HashTag Erasure Codes: From Theory to Practice** K. Kralevska, D. Gligoroski, R. E. Jensen, and H. Øverby Submitted to IEEE Transactions on Big Data
Paper 6	**Balanced Locally Repairable Codes** K. Kralevska, D. Gligoroski, and H. Øverby International Symposium on Turbo Codes and Iterative Information Processing, 2016
Paper 7	**Coded Packet Transport for Optical Packet/Burst Switched Networks** K. Kralevska, H. Øverby, and D. Gligoroski IEEE Proceedings on Global Communications Conference (GLOBECOM), pp. 1 - 6, 2015

with balanced structure (Paper 1 and Paper 2). The balanced structure ensures that all packets have the same encoding complexity. A comparison between the decoding probability with the proposed balanced codes and RLNC is given in Paper 2. If the intermediate nodes in addition to the source nodes are allowed to encode the data, then we study a case of network coding. We identified the open problem of header compression that plays a major role in practical implementations of network coding. Employing some known facts from finite fields, we suggest an algorithm called Small Set of Allowed Coefficients (SSAC) in Paper 3. Network coding is an interesting problem that is well studied with the help of graph theory. By using some concepts of network coding, the lower bound of the repair bandwidth for a single node recovery in distributed storage networks is derived. A general construction of MDS codes for any sub-packetization level for repair of a single systematic node is suggested in

Table 3.2: List of publications not included in the thesis.

Paper	Title · Author List · Conference/Journal
Paper 8	**Combining Forward Error Correction and Network Coding in Bufferless Networks: a Case Study for Optical Packet Switching** G. Biczók, Y. Chen, K. Kralevska, and H. Øverby IEEE 17th International Conference on High Performance Switching and Routing (HPSR), 2016
Paper 9	**Performance Analysis of LTE Networks with Random Linear Network Coding** T. Degefa Assefa, K. Kralevska, and Y. Jiang 39th International Convention on Information and Communication Technology, Electronics and Microelectronics, 2016
Paper 10	**Joint Balanced Source and Network Coding** K. Kralevska, H. Øverby, and D. Gligoroski 22nd Telecommunications Forum (TELFOR), Telecommunication Society, ISBN 978-1-4799-6190-0, pp. 589-592, 2014

Paper 4. The code construction from Paper 4 is further elaborated and optimized in terms of I/O in Paper 5. Later the idea of applying balanced codes as LRCs in distributed storage came. In Paper 6, we relaxed the condition of equal number of non-zero elements per column in the generator matrix. In both Paper 2 and Paper 6, we use hill-climbing search for codes with better performance. The path diversity and the secrecy features from Paper 1 inspired us to apply them in OPS/OBS networks in order to achieve survivability against link failures and non-cryptographic secrecy.

3.3 Research Answers

Here we summarize the answers to the research questions defined in Subsection 3.1.

ANS1 Paper 1 and Paper 2 answer RQ1 by introducing a new way of constructing balanced binary codes from combinatorial designs (Latin squares and Latin rectangles).

ANS2 Paper 3 answers RQ2 by proposing a novel algorithm called SSAC for practical network coding. SSAC generates the shortest header overhead by using sparse coding and properties of finite fields.

ANS3 Paper 4, Paper 5 and Paper 6 answer RQ3. Paper 4 and Paper 5 present the first MDS codes for both low-rate and high-rate regimes that provide the lowest repair bandwidth for any sub-packetization level. The practicality of

Table 3.3: List of patent applications.

Application No.	Title · Status
US14/902,251	**Network Coding over** $GF(2)$ Pending
PCT/EP2015/063337	**Coding in Galois Fields with Reduced Complexity** Pending
US 9,430,443	**Systematic Coding Technique** Granted
GB1522869.5	**Systematic Erasure Coding Technique** Pending
GB1608441.0	**Locally Repairable Erasure Codes** Pending
GB1613575.8	**Regenerating - Locally Repairable Codes** Pending
GB1616704.1	**Regenerating - Locally Repairable Codes** Pending

these codes is further investigated in Paper 5. Paper 6 solves partially the open problem of finding a general construction of LRCs with a low locality for any n and k.

ANS4 Paper 7 constitutes a first step for providing a unified view on the interactions between survivability, PLR and secrecy in OPS/OBS networks raised in RQ4.

3.4 Summary of the Results Contributing to the Thesis

This Section gives a summary of the papers included in Part II. The contributions of each paper are compared to the most relevant state-of-the-art results. The papers are presented in the order as they give answers to the research questions.

Paper 1: Balanced XOR-ed Coding
K. Kralevska, D. Gligoroski, and H. Øverby
Lecture Notes in Computer Science, vol. 8115, pp. 161-172, 2013

Encoding and decoding over $GF(2)$ are up to two orders of magnitude less energy demanding and up to one order of magnitude faster than encoding/decoding operations in higher fields [VPFH10]. Sparse codes minimize the computation time of computing any coded packet, a property that is appealing to systems where computational load-balancing is critical. These are the main motivations to seek for sparse codes only with XOR operations.

The first theoretical work by Riis in [Rii04] shows that every solvable multicast network has a linear solution over $GF(2)$. Afterwards, XOR coding has been applied in wireless networks in [KRH+08] where random coding with no predefined code construction is used.

In Paper 1, we apply the knowledge from combinatorics for code constructions by introducing a new way of constructing balanced XOR-ed codes from combinatorial designs (Latin squares and Latin rectangles). In this context, balanced means all packets are encoded with equal complexity, i.e. the number of ones in each row and each column is equal. We show that the XOR-ed codes reach the max-flow for single source multicast acyclic networks with delay. Encoding of the original data is done by a nonsingular incidence matrix obtained from a Latin rectangle, while decoding is performed by the inverse matrix of the incidence matrix. Additionally, this paper shows that balanced coding offers plausible secrecy properties. In particular, if the incidence matrix and its inverse matrix switch the roles, then an eavesdropper has to eavesdrop at least max-flow links in order to decode one original packet.

Paper 2: Families of Optimal Binary Non-MDS Erasure Codes
D. Gligoroski and K. Kralevska
IEEE Proceedings on International Symposium on Information Theory (ISIT), pp. 3150-3154, 2014

The results presented in Paper 1 are further extended in Paper 2. We use the same logic (combinatorial designs) for constructing codes as in Paper 1. First, we define families of optimal binary non-MDS erasure codes. We also introduce the metric vector of exact decoding probability as a measure for how far away a specific (n, k) code is from being an MDS code. The second contribution is a heuristic algorithm for finding those families by using hill climbing techniques over balanced XOR-ed codes. Due to the hill climbing search, those families of codes have always better decoding probability than the codes generated in a typical RLNC scenario. Finally, we show that for small values of k, the decoding probability of balanced XOR-ed codes in $GF(2)$ is very close to the decoding probability of random linear codes in $GF(4)$.

Paper 3: Minimal Header Overhead for Random Linear Network Coding
D. Gligoroski, K. Kralevska, and H. Øverby
IEEE International Conference on Communication Workshop (ICCW), pp. 680 - 685, 2015

Paper 3 presents the only algorithm in the literature called SSAC where the header length does not depend from the size of the finite field. This is achieved by applying sparse coding, using an irreducible polynomial and at least two primitive elements from a finite field.

Although the concept of sparse coding is first used in [SKFA09], there the header

length depends from the field size. Our work builds up on sparse coding, but we do not use parity-check matrices of error correcting codes. SSAC generates the header overhead by utilizing a small set $Q \subset GF(q)$ of coefficients that multiply the original data. Usually Q consists of 2 primitive elements in $GF(q)$, thus, the header length is decreased from $n \log_2 q$ to $m(1 + \log_2 n)$ bits where n is the generation size and m is the sparsity parameter. We show that the header length in SSAC does not depend on the size of the finite field where the operations are performed, i.e. it just depends on the number of combined packets. Moreover, our work is the first one that investigates the efficiency of the header compression algorithm in every intermediate node in conjunction with the number of buffered packets in that node.

Paper 4: General Sub-packetized Access-Optimal Regenerating Codes
K. Kralevska, D. Gligoroski, and H. Øverby
IEEE Communications Letters, vol. 20, issue 7, pp. 1281 - 1284, 2016

In Paper 4, we propose an algorithm for explicit construction of MDS codes that access and transfer the lowest amount of data when repairing from a single node failure for any sub-packetization level. The amount of accessed and transferred data is the same. The number of helper nodes is $n - 1$.

The algorithm presented in Paper 4 is so general that it also covers construction of access-optimal MSR codes. Compared to the construction presented in [ASVK15] where $\frac{k}{r}$ has to be an integer and the considered sub-packetization level is exclusively equal to $r^{\frac{k}{r}}$, the parameter $\frac{k}{r}$ in our work is not necessarily an integer. For instance, the code $(14, 10)$ that is deployed in the data-warehouse cluster of Facebook is out of the scope of applicability with the current proposals in [ASVK15, CHLM11, TWB14], because $\frac{k}{r} = 2.5$ is a non-integer. While the algorithm in Paper 4 constructs a $(14, 10, 13)$ code that reduces the repair bandwidth for any systematic node by 67.5% when the sub-packetization level is $r^{\lceil \frac{k}{r} \rceil} = 64$ compared to a $(14, 10)$ RS code. The presented codes are simultaneously optimal in terms of storage, reliability and repair bandwidth. We also give an algorithm for exact repair of any systematic node that is linear and highly parallelized. This means a set of $\lceil \frac{\alpha}{r} \rceil$ symbols is independently repaired first and used along with the accessed data from other helper nodes to recover the remaining symbols. The results show how the repair bandwidth decreases as the sub-packetization level increases. The lower bound of the repair bandwidth is achieved for $\alpha = r^{\lceil \frac{k}{r} \rceil}$ (MSR codes).

Paper 5: HashTag Erasure Codes: From Theory to Practice
K. Kralevska, D. Gligoroski, R. E. Jensen, and H. Øverby
Submitted to IEEE Transactions on Big Data

Paper 5 is an extension of Paper 4 where we study both the theoretical and the practical aspects of the explicit construction introduced in Paper 4. Although we first introduced the codes without a specific name, in Paper 5 we call them *HashTag*

Erasure Codes (HTECs) due to the resemblance between the hashtag sign # and the procedure of their construction. The three main contributions of Paper 5 are: an analysis of different concrete instances of HTECs, an elaboration of the correlation between the repair bandwidth and the I/Os with the sub-packetization level, and the repair bandwidth savings with HTECs even for repair of multiple failures.

We have implemented HTECs in C/C++ and performance analysis show up to 30% bandwidth savings compared to Piggyback 1 and Piggyback 2 codes [RSR13]. We also optimize HTECs in terms of the I/Os while still retaining their optimality in terms of the storage and the repair bandwidth. The authors in [RNW$^+$15] transform Product-Matrix-MSR (PM-MSR) into I/O optimal codes (which they call PM-RBT codes). Compared to HTECs that exist for any code parameters, PM-RBT codes exist only for $r \geq k - 1$. We identify the values of sub-packetization levels that give optimal overall system performance. We also show that the scheduling of the indexes in HTECs ensures a gradual increase in the number of random reads, hence no additional algorithms such as hop-and-couple [RSG$^+$14] are needed to make the reads sequential.

Additionally, HTECs are the first codes in the literature that offer bandwidth savings when recovering from multiple failures for any code parameters including the high-rate regime.

Paper 6: Balanced Locally Repairable Codes
K. Kralevska, D. Gligoroski, and H. Øverby
International Symposium on Turbo Codes and Iterative Information Processing, 2016

Paper 6 solves partially the open problem of finding a general construction of LRCs for any n and k [TPD13]. We suggest BLRCs that provide a good trade-off between the storage overhead, the repair bandwidth, MTTDL and the update complexity.

The main problem with many existing LRCs [HSX$^+$12, SAP$^+$13] is that although they reduce the size of the subset of contacted nodes, they suffer from the drawback that only a single subset of nodes enables the repair of a specific block. If a single node from that repair subset is not available, data cannot be repaired "locally" and this increases the repair cost. BLRCs address this problem and provide an efficient repair even when double failures occur. The strict requirement that the repair locality has to be a fixed small number l is relaxed for BLRCs and we allow the repair locality to be either l or $l + 1$. One of the main features of the proposed codes is that the parity blocks depend in a balanced manner from the systematic data blocks. This means that each systematic block is included in exactly w parity blocks. Additionally, BLRCs are optimal even when double failures occur and this is not the case with other LRCs. We use four metrics such as storage overhead, average repair bandwidth, MTTDL and update complexity to compare our codes with existing LRCs. An extensive reliablity analysis for calculating the MTTDL is also presented.

Paper 7: Coded Packet Transport for Optical Packet/Burst Switched Networks
K. Kralevska, H. Øverby, and D. Gligoroski
IEEE Proceedings on Global Communications Conference (GLOBECOM), pp. 1 - 6, 2015

Paper 7 provides a unified view on QoS in OPS/OBS networks. The work in Paper 7 focuses on the interactions between survivability, PLR and secrecy. The authors in [Øve04] and [VZ06] focus only on FEC codes to reduce packet loss in OPS networks. The work in [Øve08] and [ØBBT12] extends these schemes to provide 1+1 path protection. Unlike our work, none of these references considers secrecy.

We present the Coded Packet Transport (CPT) scheme, a novel transport mechanism for OPS/OBS networks that exploits the benefits of source coding with erasure codes combined with path diversity. At an OPS/OBS egress node, reconstruction of lost packets due to contentions and link/node failures is enabled by the added redundancy. Sending different subsets of non-systematic coded packets over disjoint paths between the ingress and the egress node provides an end-to-end secrecy against passive adversaries. CPT provides a non-cryptographic secrecy in OPS networks. We combine an attack technique (a combination of partially known coded text and a brute force attack) with the modern recommended levels of security (a long term security level of 128 bits) to analyze the secrecy constraints in CPT. The presented analytical models show how the QoS aspects of CPT are affected by the number of disjoint paths, the packet overhead and the packet loss rate. The number of disjoint paths and the packet overhead should be chosen so that CPT is within the operational range (the secrecy and the survivability constraints are not violated).

3.5 Concluding Remarks

As the amount of generated and stored data is exponentially growing, cost-efficient and reliable systems have become increasingly important. One way to ensure efficient reliability are erasure codes with properties as close as possible to MDS codes. Erasure codes have become particularly attractive for fault protection in storage systems. In order to have a practical deployment of new erasure coding techniques in distributed storage, several issues have to be solved. New coding techniques have to provide high resilience to failures as well as low repair bandwidth and fast recovery.

Network coding offers throughput benefits, but at the cost of adding extra overhead. Transmitting a single bit in ad hoc sensor networks is more energy consuming than performing instructions in the devices. The large amount of traffic, has made all-optical network architectures crucial for high-speed transport. OPS is a promising candidate among all-optical network architectures proposed in recent literature. We elaborate the applicability of erasure coding in OPS/OBS networks in order to provide better QoS.

The present thesis has dealt with erasure code constructions, with a particular focus on general binary and non-binary code constructions and the advantages of employing them in different networks. The overall major scientific contributions in the present thesis include:

- A novel construction of binary codes suitable for implementation on devices with limited processing and energy capacity from combinatorial designs.
- A new algorithm for header compression in network coding where the header length is 2 to 7 times shorter than the length achieved by related compression techniques.
- Code constructions and implementation of new erasure codes for large scale distributed storage systems that provide savings in the storage and network resources.
    - A novel construction of MDS erasure codes that significantly reduce both the repair bandwidth and the random I/Os during a repair of missing or otherwise unavailable data with no additional storage overhead and flexibility in the choice of parameters.
    - A code construction optimized for repair locality and update complexity by relaxing the MDS requirement.
- A unified view of QoS in OPS/OBS networks by linking survivability, packet loss rate and secrecy using erasure coding and path diversity.

3.6 Future Works

Some proposals for future works are presented in this Section.

A natural follow-up of the current work related to large-scale distributed storage is an extension of the HTEC construction to enable construction of high-rate codes that are optimal for recovery of both the systematic and the parity nodes. Several high-rate MSR codes for efficient repair of both systematic and parity nodes exist in the literature [WTB11, SAK15, YB16a]. Still for these codes, either the sub-packetization level is too large or the constructions are not explicit. An open issue is how to extend the HTEC construction to support an efficient repair of the parity nodes as well.

In Paper 4 and Paper 5, we use the work in [ASVK15] to guarantee the existence of non-zero coefficients from $\mathbf{F}_q$ so that the code is MDS. However, the lower bound of the size of the finite field is relatively big. On the other hand, in our examples we actually work with very small finite fields ($\mathbf{F}_{16}$ and $\mathbf{F}_{32}$). Recent results in [YB16b] showed that a code is access-optimal for $\alpha = r^{\lceil \frac{k}{r} \rceil}$ over any finite field $\mathbf{F}$ as long as $|\mathbf{F}| \geq r\lceil \frac{k}{r} \rceil$. Determining the lower bound of the size of the finite field for HTECs remains an open problem.

Repairing the data with regenerating nodes requires contacting $n-1$ nodes. This creates a burden in the system and the data from all $n-1$ nodes should be always available. LRCs tackle this problem by contacting only l nodes, but the improved performance comes at the expense of extra storage. An interesting research direction is how to combine the benefits from both regenerating and LRC codes.

Another research direction is the development of Error-Correcting Code (ECC) memory. The concept is similar to what we have been working on until now, but instead of recovering lost data distributed over different failure domains such as hard drives, nodes, racks and geographical locations, the concept is applied to memory chips. ECC memory offers error detection in addition to error correction. A memory error is an event that leads to the logical state of one or multiple bits being read differently from how they were last written. A memory error can lead to a machine crash or applications using corrupted data if the system does not support ECCs. Thus, ECCs memory are crucial components in each system. Since the construction of Hamming codes, only very few practical constructions have been designed and employed. Hamming codes are the most common used codes for protecting memory, although triple modular redundancy is used sometimes. Hamming codes are Single-Error-Correcting and Double-Error-Detecting (SECDED). BCH codes are another alternative for practical implementations of ECCs. They have better correcting capabilities than Hamming codes, but imply a higher latency. Thus, another research direction is a construction of low latency ECCs that offer multi-bit detection and correction in one cycle.

References

[ACLY00] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.

[ADMK05] S. Acedański, S. Deb, M. Médard, and R. Kötter. How good is random linear coding based distributed networked storage. In *NetCod*, 2005.

[Apo01] J. G. Apostolopoulos. Reliable video communication over lossy packet networks using multiple state encoding and path diversity. In *VCIP*, Proceedings of SPIE, pages 392–409, 2001.

[ASV10] N.P. Anthapadmanabhan, E. Soljanin, and S. Vishwanath. Update-efficient codes for erasure correction. In *48th Annual Allerton Conference on Communication, Control, and Computing*, pages 376–382, Sept 2010.

[ASVK15] G.K. Agarwal, B. Sasidharan, and P. Vijay Kumar. An alternate construction of an access-optimal regenerating code with optimal sub-packetization level. *National Conference on Communications (NCC)*, pages 1–6, Feb 2015.

[BBBM95] M. Blaum, J. Brady, J. Bruck, and Jai Menon. Evenodd: an efficient scheme for tolerating double disk failures in raid architectures. *IEEE Transactions on Computers*, 44(2):192–202, Feb 1995.

[Ber68] E. R. Berlekamp. *Algebraic Coding Theory.* MacGraw-Hill (New York NY), 1968.

[BGT93] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding: Turbo-codes. In *ICC*, volume 2, pages 1064–1070, 1993.

[BKW97] J. Blömer, R. Karp, and E. Welzl. The rank of sparse random matrices over finite fields. *Random Struct. Algorithms*, 10(4):407–419, July 1997.

[BLMR98] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. Technical report, May 1998.

[BN05] K. Bhattad and K.R. Narayanan. Weakly secure network coding. *Proc. First Workshop on Network Coding, Theory, and Applications (NetCod)*, 2005.

[BNH02] S. Bjørnstad, M. Nord, and D.R. Hjelme. Transparent optical protection switching scheme based on detection of polarisation fluctuations. In *Optical Fiber Communication Conference and Exhibit*, pages 433–434, Mar 2002.

[BØ11] G. Biczók and H. Øverby. Combating packet loss in ops networks: A case for network coding. In *NIK*, 2011.

[Bol79] B. Bollobas. *Graph Theory: An Introductory Course*. Springer, New York/Berlin, 1979.

[BRC60] R. C. Bose and Dwijendra K. Ray-Chaudhuri. On A class of error correcting binary group codes. *Information and Control*, 3(1):68–79, March 1960.

[CAKSAL+15] Y. Coulibaly, A. Al-Kilany, M. S. A. Latiff, G. Rouskas, S. Mandala, and M.A. Razzaque. Secure burst control packet scheme for optical burst switching networks. In *IEEE International Broadband and Photonics Conference (IBP)*, pages 86–91, April 2015.

[CC81] G. C. Clark, and J. Bibb Cain. *Error-correction coding for digital communications*. Plenum Press, 1981.

[CC84] R. Comroe and D.J. Costello. Arq schemes for data transmission in mobile radio systems. *IEEE Journal on Selected Areas in Communications*, 2(4):472–481, July 1984.

[CCW10] C.-C. Chao, C.-C. Chou, and H.-Y. Wei. Pseudo random network coding design for IEEE 802.16m enhanced multicast and broadcast service. In *VTC Spring*, pages 1–5. IEEE, 2010.

[CEG+04] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar. Row-diagonal parity for double disk failure correction. In *Proceedings of the USENIX FAST '04 Conference on File and Storage Technologies*, pages 1–14, March 2004.

[CHL11] V. R. Cadambe, C. Huang, and J. Li. Permutation code: Optimal exact-repair of a single failed node in MDS code based distributed storage systems. In *ISIT*, pages 1225–1229. IEEE, 2011.

[CHLM11] V.R. Cadambe, C. Huang, J. Li, and S. Mehrotra. Polynomial length mds codes with optimal repair in distributed storage. *Asilomar Conference on Signals, Systems and Computers*, pages 1850–1854, Nov 2011.

[CJ08] V.R. Cadambe and S.A. Jafar. Interference alignment and degrees of freedom of the k -user interference channel. *IEEE Transactions on Information Theory*, 54(8):3425–3441, Aug 2008.

[CJM10] V. R. Cadambe, S. A. Jafar, and H. Maleki. Distributed data storage with minimum storage regenerating codes - exact and functional repair are asymptotically equally efficient. *CoRR*, abs/1004.4299, 2010.

[CQY04] Y. Chen, C. Qiao, and X. Yu. Optical burst switching (obs): A new area in optical networking research. *IEEE Network Magazine*, 18:16–23, 2004.

[CR10] S. Changho and K. Ramchandran. Exact regeneration codes for distributed storage repair using interference alignment. *CoRR*, abs/1001.0107, 2010.

[CS13] J. Chen and K. W. Shum. Repairing multiple failures in the suh-ramchandran regenerating codes. In *ISIT*, pages 1441–1445, 2013.

[CV08] Y. Chen and P.K. Verma. Secure optical burst switching: Framework and research directions. *IEEE Communications Magazine*, 46(8):40–45, 2008.

[CWXQ03] Y. Chen, H. Wu, D. Xu, and C. Qiao. Performance analysis of optical burst switched node with deflection routing. In *ICC*, pages 1355–1359. IEEE, 2003.

[CY02] N. Cai and R.W. Yeung. Secure network coding. In *IEEE International Symposium on Information Theory*, pages 323–328, 2002.

[dB96] M. A. de Boer. Almost mds codes. *Des. Codes Cryptography*, 9(2):143–155, 1996.

[DGW+10] A.G. Dimakis, P.B. Godfrey, Y. Wu, M.J. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. *IEEE Transactions on Information Theory*, 56(9):4539–4551, Sept 2010.

[DJMS98] S.L. Danielsen, C. Joergensen, B. Mikkelsen, and K.E. Stubkjaer. Optical packet switched network layer without optical buffers. *Photonics Technology Letters, IEEE*, 10(6):896–898, 1998.

[DL95] S.M. Dodunekov and I.N. Landjev. On near-mds codes. *Journal of Geometry*, 54:30–43, 1995.

[DRWS11] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh. A survey on network codes for distributed storage. *Proceedings of the IEEE*, 99(3):476–489, 2011.

[DSDY13] S. H. Dau, W. Song, Z. Dong, and C. Yuen. Balanced sparsest generator matrices for mds codes. In *IEEE International Symposium on Information Theory Proceedings (ISIT)*, pages 1889–1893, July 2013.

[EM00] J.M.H. Elmirghani and H.T. Mouftah. All-optical wavelength conversion: technologies and applications in dwdm networks. *IEEE Communications Magazine*, 38(3):86–92, Mar 2000.

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

[FLP+10] D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan. Availability in globally distributed storage systems. In *9th USENIX Symposium on Operating Systems Design and Implementation, OSDI*, pages 61–74. USENIX Association, 2010.

[FLS+14] S. Feizi, D.E. Lucani, C.W. Sorensen, A. Makhdoumi, and M. Medard. Tunable sparse network coding for multicast networks. In *International Symposium on Network Coding (NetCod)*, pages 1–6, June 2014.

[FR12] M. H. Firooz and S. Roy. Data dissemination in wireless networks with network coding. *IEEE Communications Letters, Volume:17 , Issue: 5, 2013*, December 08 2012.

[FV00] A. Fumagalli and L. Valcarenghi. Ip restoration vs. wdm protection: is there an optimal choice? *IEEE Network*, 14(6):34–41, Nov 2000.

[GA02] S. Ghosh and V. Anantharam. Bufferless all-optical networking with erasure codes. In *IEEE Information Theory Workshop*, pages 19–24, Oct 2002.

[Gal63] R. G. Gallager. *Low-Density Parity-Check Codes*. The M.I.T. Press, Cambridge, MA, USA, 1963.

[GFZ03] J. Garcia-Frias and W. Zhong. Approaching shannon performance by iterative decoding of linear codes with low-density generator matrix. *IEEE Communications Letters*, 7(6):266–268, June 2003.

[GGL03] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In *ACM Symposium on Operating Systems Principles*, SOSP '03, pages 29–43. ACM, 2003.

[GHSY12] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin. On the locality of codeword symbols. *IEEE Transactions on Information Theory*, 58(11):6925–6934, 2012.

[GL03] D. W. Griffith and S. Lee. A 1+1 protection architecture for optical burst switched networks. *IEEE Journal on Selected Areas in Communications*, 21(9):1384–1398, 2003.

[GLW10] K.M. Greenan, X. Li, and J.J. Wylie. Flat xor-based erasure codes in storage systems: Constructions, efficient recovery, and tradeoffs. In *IEEE Symposium on Mass Storage Systems and Technologies (MSST)*, pages 1–14, May 2010.

[GR06] C. Gkantsidis and P. Rodriguez. Cooperative security for network coding file distribution. In *INFOCOM*. IEEE, 2006.

[GRG+98] P. Gambini, M. Renaud, C. Guillemot, F. Callegati, I. Andonovic, B. Bostica, D. Chiaroni, G. Corazza, S. L. Danielsen, P. Gravey, P. B. Hansen, M. Henry, C. Janz, A. Kloch, R. Krähenbühl, C. Raffaelli, M. Schilling, A. Talneau, and L. Zucchelli. Transparent optical packet switching: network architecture and demonstrators in the KEOPS project. *IEEE Journal on Selected Areas in Communications*, 16(7):1245–1259, 1998.

[HA00] D.K. Hunter and I. Andonovic. Approaches to optical internet packet switching. *IEEE Communications Magazine*, 38(9):116–122, Sep 2000.

[Haf05] J. L. Hafner. Weaver codes: Highly fault tolerant erasure codes for storage systems. In *FAST*. USENIX, 2005.

[HCA98] D. K. Hunter, M. C. Chia, and I. Andonovic. Buffering in optical packet switches. *Journal of Lightwave Technology*, 16(12):2081–2094, Dec 1998.

[HCL13] C. Huang, M. Chen, and J. Li. Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems. *TOS*, 9(1):3, 2013.

[HLH16] W. Halbawi, Z. Liu, and B. Hassibi. Balanced reed-solomon codes. In *IEEE International Symposium on Information Theory (ISIT)*, pages 935–939, July 2016.

[HMK+06] T. Ho, M. Médard, R. Kötter, D. R. Karger, M. Effros, J. Shi, and B. Leong. A random linear network coding approach to multicast. *IEEE Transactions on Information Theory*, 52(10):4413–4430, 2006.

[Hoc59] A. Hocquenghem. Codes correcteurs d'Erreurs. *Chiffres (Paris)*, 2:147–156, September 1959.

[HPFL08] J. Heide, M. V. Pedersen, F. H. P. Fitzek, and T. Larsen. Cautious view on network coding - from theory to practice. *Journal of Communications and Networks*, 10(4):403–411, 2008.

[HPFM11] J. Heide, M. V. Pedersen, F. H. P. Fitzek, and M. Médard. On code parameters and coding vector representation for practical RLNC. In *ICC*, pages 1–5, 2011.

[HSX+12] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin. Erasure coding in windows azure storage. In *USENIX Annual Technical Conference*, pages 15–26, 2012.

[htt] http://www.advaoptical.com/en/products/technology/dwdm.aspx. *ADVA Optical Networking*.

[HXW+10] Y. Hu, Y. Xu, X. Wang, C. Zhan, and P. Li. Cooperative recovery of distributed storage systems from multiple losses with network coding. *IEEE Journal on Selected Areas in Communications*, 28(2):268–276, February 2010.

[IDC12] IDC. Idc's digital universe study, sponsored by emc. *White Paper*, December 2012.

[KAK08] A.E. Kamal and O. Al-Kofahi. Toward an optimal 1+n protection strategy. In *Allerton Conference on Communication, Control, and Computing*, pages 162–169, Sept 2008.

[Kam06] A. E. Kamal. 1+N protection in mesh networks using network coding over p-cycles. In *GLOBECOM*, 2006.

[Kam07a] A. E. Kamal. 1+N protection against multiple link failures in mesh networks. In *ICC*, pages 2224–2229, 2007.

[Kam07b] A. E. Kamal. GMPLS-based hybrid 1+N link protection over p-cycles: Design and performance. In *GLOBECOM*, pages 2298–2303, 2007.

[Kam08] A. E. Kamal. A generalized strategy for 1+N protection. In *ICC*, pages 5155–5159, 2008.

[KBØS10] A. Kimsas, S. Bjørnstad, H. Øverby, and N. Stol. Improving performance in the opmigua hybrid network employing the network layer packet redundancy scheme. *IET Communications*, 4(3):334–342, 2010.

[KBP+12] O. Khan, R. C. Burns, J. S. Plank, W. Pierce, and C. Huang. Rethinking erasure codes for cloud file systems: minimizing I/O for recovery and degraded reads. In *FAST*, pages 20–40. USENIX Association, 2012.

[KHH+13] J. Krigslund, J. Hansen, M. Hundeboll, D. E. Lucani, and F. H. P. Fitzek. CORE: COPE with MORE in wireless meshed networks. In *VTC Spring*, pages 1–6, 2013.

[KK08] R. Kötter and F. R. Kschischang. Coding for errors and erasures in random network coding. *IEEE Transactions on Information Theory*, 54(8):3579–3591, 2008.

[KM03] R. Koetter and M. Médard. An algebraic approach to network coding. *IEEE/ACM Trans. Netw*, 11(5):782–795, 2003.

[KRH+08] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. XORs in the air: Practical wireless network coding. *IEEE/ACM Trans. Netw*, 16(3):497–510, 2008.

[KRLL11] A. E. Kamal, A. Ramamoorthy, L. Long, and S. Li. Overlay protection against link failures using network coding. *IEEE/ACM Trans. Netw*, 19(4):1071–1084, 2011.

[KSS11] A. M. Kermarrec, N. Le Scouarnec, and G. Straub. Repairing multiple failures with coordinated and adaptive regenerating codes. In *International Symposium on Network Coding*, pages 1–6, July 2011.

[Law01] E. Lawler. *Combinatorial Optimization : Networks and Matroids*. Dover Publications, 2001.

[LC83] S. Lin and D. J. Costello. *Error Control Coding: Fundamentals and Applications*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1983.

[LL14] J. Li and B. Li. Cooperative repair with minimum-storage regenerating codes for distributed storage. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 316–324, April 2014.

[LLL15] R. Li, J. Lin, and P. P. C. Lee. Enabling concurrent failure recovery for regenerating-coding-based storage systems: From theory to practice. *IEEE Transactions on Computers*, 64(7):1898–1911, July 2015.

[LMS+97] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann. Practical loss-resilient codes. In *STOC*, pages 150–159. ACM, 1997.

[LMS09] D. E. Lucani, M. Médard, and M. Stojanovic. Random linear network coding for time-division duplexing: Field size considerations. In *GLOBECOM*, pages 1–6, 2009.

[LMSS01a] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Efficient erasure correcting codes. *IEEE Transactions on Information Theory*, 47(2):569–584, Feb 2001.

[LMSS01b] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Improved low-density parity-check codes using irregular graphs. *IEEE Transactions on Information Theory*, 47(2):585–598, Feb 2001.

[LN86] R. Lidl and H. Niederreiter. *Introduction to Finite Fields and Their Applications*. Cambridge University Press, New York, NY, USA, 1986.

[LPHF14a] D. E. Lucani, M. V. Pedersen, J. Heide, and F. H. P. Fitzek. Coping with the upcoming heterogeneity in 5G communications and storage using fulcrum network codes. In *ISWCS*, pages 997–1001, 2014.

[LPHF14b] D. E. Lucani, M. V. Pedersen, J. Heide, and F. H. P. Fitzek. Fulcrum network codes: A code for fluid allocation of complexity. *CoRR*, abs/1404.6620, 2014.

[LR10] S. Li and A. Ramamoorthy. Improved compression of network coding vectors using erasure decoding and list decoding. *IEEE Communications Letters*, 14(8):749–751, 2010.

[Lub02] M. Luby. LT codes. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2002.

[LWLZ10] Z. Liu, C. Wu, B. Li, and S. Zhao. UUSee: Large-scale operational on-demand streaming with random network coding. In *IEEE INFOCOM*, pages 2070–2078, 2010.

[LY82] S. Lin and P.S. Yu. A hybrid arq scheme with parity retransmission for error control of satellite channels. *IEEE Transactions on Communications*, 30(7):1701–1719, July 1982.

[LYC03] S. Y. R. Li, R. W. Yeung, and N. Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49, 2003.

[MDXA10] E. D. Manley, J.S. Deogun, L. Xu, and D. R. Alexander. All-optical network coding. *IEEE/OSA Journal of Optical Communications and Networking*, 2(4):175–191, April 2010.

[Men] K. Menger. Zur allgemeinen kurventheorie.

[MFHH02] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A tiny AGgregation service for ad-hoc sensor networks. In *OSDI*, pages 1–16, 2002.

[MG08] R.C. Menendez and J.W. Gannet. Efficient, fault-tolerant all-optical multicast networks via network coding. In *Optical Fiber communication/National Fiber Optic Engineers Conference*, pages 1–3, Feb 2008.

[MN95] D. J. C. MacKay and R. M. Neal. Good codes based on very sparse matrices. *Lecture Notes in Computer Science*, 1025, 1995.

[MN97] D. J. C. MacKay and R. M. Neal. Near shannon limit performance of low density parity check codes. *Electronics Letters*, 33(6):457–458, Mar 1997.

[MNS14] T. Miyamae, T. Nakao, and K. Shiozawa. Erasure code with shingled local parity groups for efficient recovery from multiple disk failures. In *10th Workshop on Hot Topics in System Dependability (HotDep)*. USENIX Association, 2014.

[MS78] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-holland Publishing Company, 2nd edition, 1978.

[MS12] M. Médard and A. Sprintson. *Network coding, Fundamentals and Applications*. 2012.

[NBG08] M. Nord, S. Bjørnstad, and C. M. Gauger. OPS or OBS in the core network?- A comparison of optical packet- and optical burst switching, 2008.

[NNC10] K. Nguyen, T. P. Nguyen, and S.-C. S. Cheung. Video streaming with network coding. *Signal Processing Systems*, 59(3):319–333, 2010.

[NZ02] T. Nguyen and A. Zakhor. Distributed video streaming with forward error correction, 2002.

[NZ03] T. Nguyen and A. Zakhor. Path diversity with forward error correction (pdf) system for packet switched networks. In *INFOCOM*, volume 1, pages 663–672 vol.1, March 2003.

[ØBBT12] H. Øverby, G. Biczók, P. Babarczi, and J. Tapolcai. Cost comparison of 1+1 path protection schemes: A case for coding. In *IEEE International Conference on Communications (ICC)*, 2012.

[OD11] F. E. Oggier and A. Datta. Self-repairing homomorphic codes for distributed storage systems. In *INFOCOM*, pages 1215–1223, 2011.

[OLV+12] P. F. Oliveira, L. Lima, T. T. V. Vinhoza, J. Barros, and M. Médard. Coding for trusted storage in untrusted networks. *IEEE Transactions on Information Forensics and Security*, 7(6):1890–1899, 2012.

[OSHT01] M.J. O'Mahony, D. Simeonidou, D.K. Hunter, and A. Tzanakaki. The application of optical packet switching in future communication networks. *IEEE Communications Magazine*, 39(3):128–135, Mar 2001.

[Øve04] H. Øverby. Network layer packet redundancy in optical packet switched networks. *Opt. Express*, 12(20):4881–4895, Oct 2004.

[Øve08] H. Øverby. Combined study on survivability and performance in optical packet switched networks. *J. Opt. Netw.*, 7(4):294–309, Apr 2008.

[PB61] W. W. Peterson and D. T. Brown. Cyclic codes for error detection. *Proceedings of the IRE*, 49(1):228–235, Jan 1961.

[PDC13] D.S. Papailiopoulos, A.G. Dimakis, and V.R. Cadambe. Repair optimal erasure codes through hadamard designs. *IEEE Transactions on Information Theory*, 59(5):3021–3037, May 2013.

[PFS05] P. Pakzad, C. Fragouli, and A. Shokrollahi. Coding schemes for line networks. In *Proc. ISIT*, 2005.

[PGK88] D. A. Patterson, G. Gibson, and R. H. Katz. A case for redundant arrays of inexpensive disks (RAID). In *ACM SIGMOD International Conference on Management of Data*, pages 109–116, May 26–28 1988.

[PJBM+16] L. Pamies-Juarez, F. Blagojević, R. Mateescu, C. Gyuot, E. E. Gad, and Z. Bandić. Opening the chrysalis: On the real repair performance of msr codes. In *USENIX Conference on File and Storage Technologies (FAST)*, pages 81–94, February 2016.

[Pla05] J. S. Plank. Erasure codes for storage applications. Tutorial Slides, presented at *FAST-2005: 4th Usenix Conference on File and Storage Technologies*, 2005.

[PLD+12] D.S. Papailiopoulos, J. Luo, A.G. Dimakis, C. Huang, and J. Li. Simple regenerating codes: Network coding for cloud storage. In *INFOCOM*, pages 2801–2805, March 2012.

[QY99] C. Qiao and M. Yoo. Optical burst switching (obs) - a new paradigm for an optical internet. *Journal of High Speed Networks*, 8:69–84, 1999.

[Rii04] S. Riis. Linear versus nonlinear boolean functions in network flow. *CISS*, 2004.

[RNW+15] K. V. Rashmi, Preetum Nakkiran, Jingyan Wang, Nihar B. Shah, and Kannan Ramchandran. Having your cake and eating it too: Jointly optimal erasure codes for I/O, storage, and network-bandwidth. In Jiri Schindler and Erez Zadok, editors, *FAST*, pages 81–94. USENIX Association, 2015.

[RS60] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, June 1960.

[RS02] R. Ramaswami and K. N. Sivarajan. *Optical Networks: A Practical Perspective*. Second edition, 2002.

[RSG+14] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran. A "hitchhiker's" guide to fast and efficient data reconstruction in erasure-coded data centers. In *SIGCOMM*, pages 331–342, 2014.

[RSK10] K. V. Rashmi, N. B. Shah, and P. V. Kumar. Network coding. *Resonance*, 15(7):604–621, 2010.

[RSK11] K. V. Rashmi, N. B. Shah, and P. V. Kumar. Optimal exact-regenerating codes for distributed storage at the msr and mbr points via a product-matrix construction. *IEEE Transactions on Information Theory*, 57(8):5227–5239, Aug 2011.

[RSR13] K. V. Rashmi, N. B. Shah, and K. Ramchandran. A piggybacking design framework for read-and download-efficient distributed storage codes. *CoRR*, abs/1302.5872, 2013.

[RSU01] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Transactions on Information Theory*, 47(2):619–637, Feb 2001.

[SAK15] B. Sasidharan, G. K. Agarwal, and P. V. Kumar. A high-rate MSR code with polynomial sub-packetization level. *CoRR*, abs/1501.06662, 2015.

[SAP+13] M. Sathiamoorthy, M. Asteris, D. S. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur. XORing elephants: Novel erasure codes for big data. *PVLDB*, 6(5):325–336, 2013.

[Sho06] A. Shokrollahi. Raptor codes. *IEEE Transactions on Information Theory*, 52(6):2551–2567, 2006.

[Sil12] D. Silva. Minimum-overhead network coding in the short packet regime. In *International Symposium on Network Coding (NetCod)*, pages 173–178, June 2012.

[Sin64] R. Singleton. Maximum distance q -nary codes. *IEEE Transactions on Information Theory*, 10(2):116–118, Apr 1964.

[SKFA09] M. J. Siavoshani, L. Keller, C. Fragouli, and K. J. Argyraki. Compressed network coding vectors. In *ISIT*, pages 109–113, 2009.

[SKRC10] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *IEEE Symposium on Mass Storage Systems and Technologies (MSST)*, pages 1–10, 2010.

[SMS12] N. Sreenath, K. Muthuraj, and P. Sivasubramanian. Secure optical internet: Attack detection and prevention mechanism. In *International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*, pages 1009–1012, March 2012.

[SRM02] L. H. Sahasrabuddhe, S. Ramamurthy, and B. Mukherjee. Fault management in IP-over-WDM networks: WDM protection versus IP restoration. *IEEE Journal on Selected Areas in Communications*, 20(1):21–33, 2002.

[TCBOF11] O. Trullols-Cruces, J. M. Barcelo-Ordinas, and M. Fiore. Exact decoding probability under random linear network coding. *IEEE Communications Letters*, 15(1):67–69, January 2011.

[TF12] N. Thomos and P. Frossard. Toward one symbol network coding vectors. *IEEE Communications Letters*, 16(11):1860–1863, 2012.

[TPD13] I. Tamo, D. S. Papailiopoulos, and A. G. Dimakis. Optimal locally repairable codes and connections to matroid theory. *CoRR*, abs/1301.7693, 2013.

[Tur99] J. S. Turner. Terabit burst switching. *J. High Speed Networks*, 8(1):3–16, 1999.

[TWB13] I. Tamo, Z. Wang, and J. Bruck. Zigzag codes: MDS array codes with optimal rebuilding. *IEEE Transactions on Information Theory*, 59(3):1597–1616, 2013.

[TWB14] I. Tamo, Zhiying Wang, and J. Bruck. Access versus bandwidth in codes for storage. *IEEE Transactions on Information Theory*, 60(4):2028–2037, April 2014.

[VB09] F. Vieira and J. Barros. Network coding multicast in satellite networks. In *Next Generation Internet Networks*, pages 1–6, July 2009.

[VCR00] S. Verma, H. Chaskar, and R. Ravikanth. Optical burst switching: a viable solution for terabit ip backbone. *IEEE Network*, 14(6):48–53, Nov 2000.

[VJS02] V.M. Vokkarane, J.P. Jue, and S. Sitaraman. Burst segmentation: an approach for reducing packet loss in optical burst switched networks. In *IEEE International Conference on Communications (ICC)*, volume 5, pages 2673–2677, May 2002.

[VPFH10] P. Vingelmann, M. V. Pedersen, F. H. P. Fitzek, and J. Heide. Multimedia distribution using network coding on the iphone platform. In *ACM Multimedia Workshop on Mobile Cloud Media Computing*, MCMC '10, pages 3–6, 2010.

[VZ06] V.M. Vokkarane and Q. Zhang. Forward redundancy: a loss recovery mechanism for optical burst-switched networks. In *International Conference on Wireless and Optical Communications Networks*, pages 5–10, 2006.

[WD09] Y. Wu and A. G. Dimakis. Reducing repair traffic for erasure coding-based storage via interference alignment. In *ISIT*, pages 2276–2280, 2009.

[WK02] H. Weatherspoon and J. Kubiatowicz. Erasure coding vs. replication: A quantitative comparison. In *International Workshop on Peer-to-Peer Systems (IPTPS), LNCS*, volume 1, 2002.

[WTB11] Z. Wang, I. Tamo, and J. Bruck. On codes for optimal rebuilding access. In *Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1374–1381, Sept 2011.

[WWX10] X. Wang, J. Wang, and Y. Xu. Data dissemination in wireless sensor networks with network coding. *EURASIP J. Wireless Comm. and Networking*, 2010.

[WXHO10] X. Wang, Y. Xu, Y. Hu, and K. Ou. Mfr: Multi-loss flexible recovery in distributed storage systems. In *IEEE International Conference on Communications (ICC)*, pages 1–5, May 2010.

[YB16a] M. Ye and A. Barg. Explicit constructions of high-rate MDS array codes with optimal repair bandwidth. *CoRR*, abs/1604.00454, 2016.

[YB16b] M. Ye and A. Barg. Explicit constructions of optimal-access MDS codes with nearly optimal sub-packetization. *CoRR*, abs/1605.08630, 2016.

[YKSC01] S.-W. Yuk, M.-G. Kang, B.-C. Shin, and D.-H. Cho. An adaptive redundancy control method for erasure-code-based real-time data transmission over the internet. *IEEE Transactions on Multimedia*, 3(3):366–374, Sep 2001.

[YSH+98] Y. Yamada, K. Sasayama, K. Habara, A. Misawa, M. Tsukada, T. Matsunaga, and K. Yukimatsu. Optical output buffered ATM switch prototype based on FRONTIERNET architecture. *IEEE Journal on Selected Areas in Communications*, 16(7):1298–1308, 1998.

[ZS00] D. Zhou and S.S. Subramaniam. Survivability in optical networks. *IEEE Network*, 14(6):16–23, Nov 2000.

Part II

Included Papers

Paper 1

Balanced XOR-ed Coding

Katina Kralevska, Danilo Gligoroski, and Harald Øverby
Lecture Notes in Computer Science, vol. 8115, pp. 161-172, 2013

Balanced XOR-ed Coding

Katina Kralevska, Danilo Gligoroski, and Harald Øverby
Department of Telematics; Faculty of Information Technology, Mathematics and Electrical Engineering; Norwegian University of Science and Technology, Trondheim, Norway,
Email: {katinak, danilog, haraldov}@item.ntnu.no

Abstract

This paper presents a construction of codes over $GF(2)$ which reach the max-flow for single source multicast acyclic networks with delay. The coding is always a bitwise XOR of packets with equal lengths, and is based on highly symmetrical and balanced designs. For certain setups and parameters, our approach offers additional plausible security properties: an adversary needs to eavesdrop at least max-flow links in order to decode at least one original packet.

Keywords – XOR coding, $GF(2)$, Latin squares, Latin rectangles

I. INTRODUCTION

Encoding and decoding over $GF(2)$ is more energy efficient than encoding and decoding in any other larger field. Recent studies concerning several new techniques in network coding [1] (Linear Network Coding (LNC) [10], [12] and Random Linear Network Coding (RLNC) [6]) confirmed that encoding and decoding over $GF(2)$ are up to two orders of magnitude less energy demanding and up to one order of magnitude faster than the encoding/decoding operations in larger fields [14], [18], [20].

The high computational complexity of packet encoding and decoding over large finite fields and its high energy cost which makes it unsuitable for practical implementation are the main motivation to seek for coding techniques only with XOR operations. The first theoretical work was done by Riis in [16] who showed that every solvable multicast network has a linear solution over $GF(2)$. Afterwards, XOR coding in wireless networks was presented in [9], where the main rule is that a node can XOR n packets together only if the next hop has all $n-1$ packets. A more general network coding problem which is called index coding is considered in [15], [17]. In [15] the authors address the coding problem by proposing coding over $GF(2)$. The encoding scheme is based on bitwise XORing by adding redundant bits, and the decoding scheme is based on a simple but bit after bit sequential back substitution method.

The main contribution of our work is a construction of codes over $GF(2)$ by using combinatorial designs (Latin squares and Latin rectangles) [4]. Its lower computation and energy cost makes it suitable for practical implementation on devices with limited processing and energy capacity like mobile phones and wireless sensors. We will illustrate the construction of codes by the following simple example.

Example 1: We use the following strategy (Fig. 1): the source s performs bitwise XOR of packets with equal length based on the incidence matrix of a Latin rectangle L. Each column of L represents a combination of source packets x_i, $i = 1, \ldots, 4$, in a coded packet c_i, $i = 1, \ldots, 4$. In the first phase, the packets c_1 and c_2 are sent, and in the second phase, the packets c_3 and c_4 are sent. The intermediate nodes u_1 and u_2 forward the coded packets to the sink nodes t_1 and t_2 which decode the packets by using the inverse matrix of the incidence matrix of L. The sink nodes need only to know the combination of source packets in each received packet. Note that the max-flow in the network is achieved.
Routinely as in other coding approaches, this information is included in the header of each coded packet. Since in this paper we use diversity coding performed just by the source nodes, there is no need for updating the coefficients in the header at each intermediate node. The length of the prepended header vector is negligible compared to the length of the packet.

The construction of our codes was not motivated by security issues, therefore the security is not the main goal in this paper. However, it turns out that for certain setups and parameters, our approach offers

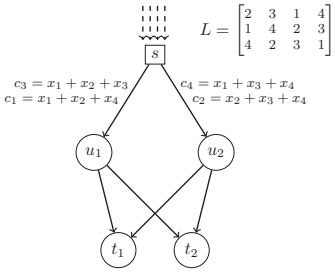


Figure 1: An example of balanced XOR coding where the source sends combinations of source packets (combined as the column of the Latin rectangle). The intermediate nodes just forward the data to the sink nodes.

additional plausible security properties. The plausible security properties that accompany our approach are not based on hard mathematical problems in modern cryptology (for example factoring of large integers or discrete logarithm problems or on the Shamir's secret sharing algorithm). We show that if an eavesdropper wants to reconstruct at least one original packet, then the number of eavesdropped links should be equal to the max-flow of the network. Bhattad and al. [2] make similar observations when network coding is implemented so that a weekly secure network coding is achieved.

The rest of the paper is organized as follows: Section II presents the notation and the mathematical background that are used in the following sections. The construction of codes is presented in Section III. Section IV illustrates the security features of our approach. Sections V concludes the paper.

II. Notation and Mathematical Background

We define a communication network as a tuple $N = (V, E, S, T)$ that consists of:

- a finite directed acyclic multigraph $G = (V, E)$ where V is the set of vertices and E is the set of edges,
- a set $S \subset V$ of sources,
- a set $T \subset V$ of sink nodes.

Assume that vertex $s \in S$ sends n source packets to vertex $t \in T$ over disjoint paths. A minimal cut separating s and t is a cut of the smallest cardinality denoted as mincut(s, t). The packets are sent in several time slots, i.e., phases denoted as p. The maximum number of packets that can be sent in a phase from s to t is denoted as maxflow(t). The Max-Flow Min-Cut Theorem [11] indicates that mincut$(s, t) =$ maxflow(t). The multicast capacity, i.e., the maximum rate at which s can transfer information to the sink nodes, cannot exceed the capacity of any cut separating s from the sink nodes. A network is solvable when the sink nodes are able to deduce the original packets with decoding operations. If the network is solvable with linear operations we say that the network is linearly solvable.

A. XOR-ed coding

First we recall that in [16], Riis showed that every solvable multicast network has a linear solution over $GF(2)$ in some vector dimension. The essence of his proof relies on the fact that any two finite fields with the same cardinality are isomorphic. Thus, instead of working in a finite field $GF(2^n)$ for which the conditions of the linear-code multicast (LCM) theorem [12, Th. 5.1] are met, he showed that it is possible to work in the isomorphic vector space $GF(2)^n$ that is an extension field over the prime field $GF(2)$. We formalize the work in the vector space $GF(2)^n$ with the following:

Definition 1: A XOR-ed coding is a coding that is realized exclusively by bitwise XOR operations between packets with equal length. Hence, it is a parallel bitwise linear transformation of n source bits $x = (x_1, \ldots, x_n)$ by a $n \times n$ nonsingular matrix K, i.e., $y = K \cdot x$.

In [16] it was also shown that there are simple network topologies where encoding in $GF(2)$ cannot reach the network capacity with the original bandwidth or by sending data in just one phase. However, it was shown that the network capacity by XOR-ed coding can be achieved either by increasing the bandwidth or the number of phases so that they match the dimension of the extended vector space $GF(2)^n$. In this paper we take the approach to send data in several phases p instead of increasing the bandwidth.

Theorem 1: For any linearly solvable network topology with maxflow$(t_1) > 1$, the sufficient condition for a single sink t_1 to reach its capacity in each of p phases by XOR-ed coding is to receive n linearly independent packets $x = (x_1, \ldots, x_n)$, where $n = p \times$ maxflow(t_1).

Proof: Assume that the network topology is linearly solvable. That means there exists a vector space $GF(2)^n$ where we can encode every n source bits with a bijective function K, i.e., $y = K \cdot x$. Having in mind that the source s succeeds to send n encoded packets to t_1 in p phases, and the max-flow in the network is maxflow$(t_1) > 1$, we have that $n = p \times$ maxflow(t_1) and the sink t_1 receives n packets after p phases via maxflow(t_1) disjoint paths. In order to have a successful recovery of the initial n packets, the received packets should be linearly independent. ■

Based on Theorem 1 we can prove the following:

Theorem 2: For any linearly solvable network topology and for any two sinks $T = \{t_1, t_2\}$ that have maxflow$(t) =$ maxflow$(t_1) =$ maxflow(t_2), there always exists a XOR-ed coding for $n = p \times$ maxflow(t) packets that achieves the multicast capacity in each of p phases.

Proof: For the sink t_1 we apply Theorem 1 and find one XOR-ed coding that achieves the capacity in each of p phases. Let us denote by $U_1 = \{u_{1,i}|$ there is an edge $(u_{1,i}, t_1) \in E\}$ the nodes that are directly connected and send packets to the sink node t_1. We have that $|U_1| =$ maxflow(t), and the set of n packets is partitioned in maxflow(t) disjoint subsets $Y_{1,1}, \ldots, Y_{1,\text{maxflow}(t)}$ each of them having p packets. The subset $Y_{1,i}$ comes from the node u_i, $i = 1, \ldots,$ maxflow(t).

The set $U_2 = \{u_{2,i}|$there is an edge$(u_{2,i}, t_2) \in E\}$ is a set of nodes that are directly connected and send packets to the sink node t_2. We denote the intersection between the sets of nodes U_1 and U_2 as $U_{1,2} = U_1 \bigcap U_2$. The following three situations are considered:

1) There are no mutual nodes that send packets to both sinks t_1 and t_2, i.e., $U_{1,2} = \emptyset$. In that case find one partition of the set of n packets in maxflow(t) disjoint subsets $\Gamma_1 = \{Y_{2,1} \ldots, Y_{2,\text{maxflow}(t)}\}$ each of them having p packets. The sets of packets $Y_{2,j}$ are delivered to the sink t_2 via the node $u_{2,j}$, $j = 1, \ldots,$ maxflow(t). The multicast capacity for the sink t_2 is achieved in each of p phases.
2) There are nodes that send packets to both sinks t_1 and t_2, i.e., $U_{1,2} = \{u_{(1,2)_{\nu_1}}, \ldots, u_{(1,2)_{\nu_k}}\}$. Denote the nodes that are in $U_2 \backslash U_1 = \{u_{2_{\nu_1}}, \ldots$ $\ldots, u_{2_{\nu_{\text{maxflow}(t)-k}}}\}$. In that case, the sink t_2 receives from the nodes in $U_{1,2}$ the same packets that are delivered to the sink t_1. The number of the remaining packets that have to be delivered to t_2 is exactly $p \times (\text{maxflow}(t) - k)$. Find one partition of maxflow$(t) - k$ disjoint subsets $\Gamma_2 = \{Y_{2,1} \ldots, Y_{2,\text{maxflow}(t)-k}\}$ each of them having p packets. The sets of packets $Y_{2,j}$ are delivered to the sink t_2 via the node u_{2,ν_j}, $j = 1, \ldots,$ maxflow$(t) - k$. The multicast capacity for the sink t_2 is achieved in each of p phases.
3) All the nodes that send packets to the sink t_1, send packets to the sink t_2 as well, i.e., $U_{1,2} = U_1 \bigcap U_2 = U_1$. In that case, the sink t_2 receives from the nodes in $U_{1,2}$ the same packets that are delivered to the sink t_1. The multicast capacity for the sink t_2 is achieved in each of p phases.

■

Note that the proof of Theorem 2 is similar to the work by Jaggi et al. [8] where they discuss a construction of general codes using simple algorithms.

As a consequence of Theorems 1 and 2 we can post the following:

Theorem 3: For any linearly solvable network topology and for any set of N sinks $T = \{t_1, \ldots, t_N\}$ that have maxflow$(t) =$ maxflow$(t_1) = \ldots =$ maxflow(t_N), there always exists a XOR-ed coding for

$n = p \times \text{maxflow}(t)$ packets that achieves the multicast capacity in each of p phases.

Proof: (*Sketch*) First, we recall the construction of generic linear codes presented in the LCM theorem in [12, Th. 5.1]. Second, we use the transformation to equivalent codes over $GF(2)^n$ as it was shown in [16]. Then, the proof is a straightforward application of the mathematical induction by the number of sinks N. Let us suppose that the claim of the theorem is correct for $N-1$ sinks. By adding a new N-th sink we consider again three possible situations as in Theorem 2. ■

III. Construction of XOR-ed Coding

In this section we describe the construction of codes over $GF(2)$. Instead of working with completely random binary matrices, in the remaining part of this paper we work with nonsingular binary matrices that have some specific structure related to randomly generated Latin square or Latin rectangle. We do not reduce the space of possible random linear network encoding schemes, since the number of Latin squares and Latin rectangles of order n increases proportionally with factorial of n. Therefore, in our approach we have virtually an endless repository of encoding schemes that have the benefits from both worlds: they are randomly generated, but they have a certain structure and offer plausible security properties.

In order to introduce our approach, we briefly use several definitions that the reader can find in [19] and [3].

Definition 2: A Latin square of order n with entries from an n-set X is an $n \times n$ array L in which every cell contains an element of X such that every row of L is a permutation of X and every column of L is a permutation of X.

Definition 3: A $k \times n$ Latin rectangle is a $k \times n$ array (where $k \leq n$) in which each cell contains a single symbol from an n-set X, such that each symbol occurs exactly once in each row and at most once in each column.

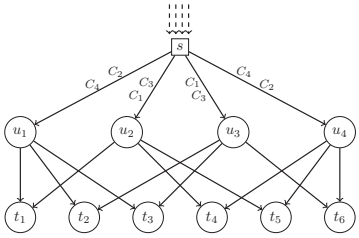


Figure 2: A 4-dimensional binary linear multicast in a single source multicast network with delay

For generating a Latin square, one can always start with a permutation of n elements that is a trivial $1 \times n$ Latin rectangle and can use the old Hall's marriage theorem [5] to construct new rows until the whole Latin square is completed. However, this approach does not guarantee that the generated Latin squares are chosen uniformly at random. In order to generate Latin squares of order n that are chosen uniformly at random we use the algorithm of Jacobsen and Matthews [7]. Further, in our approach we sometimes split the Latin square into two Latin rectangles (upper and lower), and work with the algebraic objects (matrices or block designs) that are related to either the upper or the lower Latin rectangle.

As a convention, throughout this paper, the number of packets n that are sent from the source is equal to the number of columns in the Latin square or Latin rectangle.

Example 2: As shown in Fig. 2, we assume that the source wants to send four packets $x_1, \ldots, x_4$ to the sink nodes, and that each sink node has maxflow(t_k) = 2, ($k = 1, \ldots, 6$). The sink nodes receive data from different pair of intermediate nodes, u_i, ($i = 1, \ldots, 4$). Our aim is all six sink nodes to be able to reconstruct the source packets that are exclusively coded in $GF(2)$.

Let us take the following Latin square and split it into two Latin rectangles:

$$L = \left[\begin{array}{cccc} 2 & 4 & 1 & 3 \\ 1 & 3 & 2 & 4 \\ 3 & 2 & 4 & 1 \\ \hline 4 & 1 & 3 & 2 \end{array}\right].$$

Each column from the 3×4 upper Latin rectangle represents a combination of source packets in a coded packet c_i, $i = 1, \ldots, 4$. Using the incidence matrix M of the Latin rectangle the source computes the coded packets.

Definition 4: Let (X, A) be a design where $X = \{x_1, \ldots, x_v\}$ and $A = \{A_1, \ldots, A_b\}$. The incidence matrix of (X, A) is the $v \times b$ 0-1 matrix $M = (m_{i,j})$ defined by the rule $m_{i,j} = \begin{cases} 1, & \text{if } x_i \in A_j, \\ 0, & \text{if } x_i \notin A_j. \end{cases}$

Proposition 1: The incidence matrix $M = (m_{i,j})$ of any Latin rectangle with dimensions $k \times n$ is balanced matrix with k ones in each row and each column.

Proof: From the definition of the incidence matrix it follows that the number of ones in each row is equal to the number of elements k in each column of the Latin rectangle. On the other hand, since each row of the Latin rectangle is a permutation of n elements, and there are no elements that occur twice in each column, the number of ones in each column can be neither less nor larger than k. ■

Note 1: The incidence matrix M of a $k \times n$ Latin rectangle is always balanced. However, the inverse matrix of the incidence matrix M^{-1} is not always balanced.

Proposition 2: The necessary condition an incidence matrix $M = (m_{i,j})$ of a $k \times n$ Latin rectangle to be nonsingular in $GF(2)$ is k to be odd, i.e., $k = 2l + 1$.

Proof: Assume that k is even, i.e., $k = 2l$. Recall that a matrix M is nonsingular in $GF(2)$ if and only if its determinant is 1 (or it is singular if and only if its determinant is 0). Recall further the Leibniz formula for the determinant of an $n \times n$ matrix M: $det(M) = \sum_{\sigma \in S_n} sgn(\sigma) \prod_{i=1}^{n} m_{i,\sigma_i}$, where the sum is computed over all elements of the symmetric group of n elements S_n, i.e., over all permutations $\sigma \in S_n$, and $sgn(\sigma)$ is the signature (or the parity of the permutation) whose value is $+1$ or -1. The elements m_{i,σ_i} are the elements $m_{i,j}$ of the matrix M where the value for the index $j = \sigma_i$ is determined as the i–th element of the permutation σ.

If $k = 2l$ is even, from Proposition 1 and from the fact that operations are performed in $GF(2)$, it follows that every summand in the Leibniz formula gives an even number of nonzero products, thus the final sum must be even, i.e., the determinant in $GF(2)$ is 0. ■

The corresponding 4×4 incidence matrix of the Latin rectangle in Example 2 is nonsingular in $GF(2)$ (Proposition 2). M is represented as

$$M = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}.$$

A direct consequence from Theorem 1 is the following:

Corollary 1: A sink node $t \in T$ with maxflow(t) can receive n source packets, encoded with the incidence matrix of a $k \times n$ Latin rectangle in $GF(2)$, in $p = \lceil \frac{n}{\text{maxflow}(t)} \rceil$ phases. In each phase the sink node reaches its maxflow(t).

Following Corollary 1 the number of phases in which packets are sent depends from the total number of packets and maxflow(t_k).

Using M the source computes the vector of coded packets as

$$\mathbf{c} = M\mathbf{x} = [c_1, c_2, c_3, c_4]^\top$$

TABLE I: Description of receiving coded packets in each phase at the sink nodes

	t_1	t_2	t_3	t_4	t_5	t_6
First phase	C_4, C_1	C_4, C_3	C_4, C_3	C_1, C_2	C_1, C_2	C_3, C_2
Second phase	C_2, C_3	C_2, C_1	C_2, C_1	C_3, C_4	C_3, C_4	C_1, C_4

where $\mathbf{x} = [x_1, x_2, x_3, x_4]^\top$ is a vector of the source packets. The coded packets are XOR-ed combinations of the source packets, i.e.,

$$c_1 = x_1 \oplus x_2 \oplus x_3,$$
$$c_2 = x_2 \oplus x_3 \oplus x_4,$$
$$c_3 = x_1 \oplus x_2 \oplus x_4,$$
$$c_4 = x_1 \oplus x_3 \oplus x_4.$$

The source further prepends the information from the incidence matrix to each of the coded packets. The vector of packets that are sent becomes as follows: $\mathbf{C} = \{(1, 2, 3, c_1), (2, 3, 4, c_2), (1, 2, 4, c_3), (1, 3, 4, c_4)\}$ $= \{C_1, C_2, C_3, C_4\}$. The sink nodes receive in each phase a pair of different packets as shown in Table I. Their buffer should be large enough to store the received packets C_i, $i = 1, \ldots, 4$. The decoding at the sink nodes is performed by M^{-1}. Each sink node computes M^{-1} from the prepended indexes. The original packets x_i, $i = 1, \ldots, 4$, are reconstructed as $\mathbf{x} = M^{-1}\mathbf{c}$. Note that although our approach is similar to [16], we use a systematic selection of the encoding functions and we do not send plain packets on the disjoint paths.

IV. Additional Plausible Security Properties of the Balanced XOR-ed Coding

The work with incidence matrices related to randomly generated Latin rectangles is actually a work with balanced block designs. However, as we noted in Note 1, it is not necessary both the incidence matrix and its inverse matrix to be completely balanced. If we are interested in the complexity of decoding and the security issues when an adversary can successfully decode some sniffed packets, then the easiest way to address these issues is to give equal level of security to all encoded packets. In our approach this can be easily achieved by switching the roles of the incidence matrix and its inverse matrix: the encoding of the source packets is done with the inverse matrix of the incidence matrix and decoding of the coded packets is done with the incidence matrix. By applying this approach, decoding of any of the source packets requires an equal number of coded packets.

Corollary 2: For each value of maxflow(t) and a number of source packets n which is multiple of maxflow(t), there exists a Latin rectangle with $n-1$ or $n-2$ rows and its incidence matrix can be used for decoding.

Due to Proposition 2, when n is even the necessary requirement for a nonsingular incidence matrix is the Latin rectangle to have $n-1$ rows. When n is odd the necessary requirement for a nonsingular incidence matrix is the Latin rectangle to have $n-2$ rows.

Theorem 4: When decoding is performed with the incidence matrix from Corollary 2, any eavesdropper needs to listen at least maxflow(t) links in order to decode at least one source packet.

Proof: Assume that an adversary eavesdrops maxflow$(t) - 1$ links. Since the incidence matrix used for decoding is related to a Latin rectangle with $n-1$ or $n-2$ rows, eavesdropping "just" maxflow$(t) - 1$ links is not sufficient for the adversary to receive at least one subset of $n-1$ or $n-2$ packets from which he/she can decode at least one original packet. ■

Another remark that can be given about our approach is that the number of XOR operations between different packets (both in the source and in the sink nodes) is relatively high. We can address that remark by using Latin rectangles with smaller number of rows as a trade-off between the number of encoding/decoding operations and the ability of an adversary to decode a source packet. Namely, the encoding and decoding

efforts at the source and sink node are the highest when encoding and decoding requires $n-1$ or $n-2$ packets. In order to decrease the number of operations at the nodes, the Latin rectangle should have $k \leq n-2$ rows. However, we are interested to reduce the number k without reducing the number of links that have to be listened by an eavesdropper in order to decode at least one original packet. The following theorem gives the necessary and sufficient condition for that to happen:

Theorem 5: Let the coding be done by M^{-1} obtained from a Latin rectangle $L_{k\times n}$ of size $k \times n$, where $k \leq n-2$. Further, assume that the transfer is done by sending n packets from s to t in $p = \lceil \frac{n}{\text{maxflow}(t)} \rceil$ phases on maxflow(t) disjoint paths and let the sets of indexes of the packets sent via i-th disjoint path are denoted by $S_i, i = 1, \ldots, \text{maxflow}(t)$. A necessary and sufficient condition for an eavesdropper to need to listen at least maxflow(t) links in order to decode at least one original packet is:

$$\forall j \in \{1, \ldots, n\}, \forall i \in \{1, \ldots, \text{maxflow}(t)\} : L_{k,j} \cap S_i \neq \emptyset, \tag{1}$$

where $L_{k,j}, j \in \{1, \ldots, n\}$ is the set of elements in the j-th column of the Latin rectangle $L_{k\times n}$.

Proof: To show that the condition (1) is necessary assume that an eavesdropper needs maxflow$(t)-1$ links in order to decode one original packet x_l, and let us denote by S_m the set of indexes of the packets sent via the disjoint path that was not listened by the eavesdropper. This means that for the l-th column $L_{k,l}$ of the Latin rectangle $L_{k\times n}$: $L_{k,l} \cap S_m = \emptyset$ which violates the condition (1).

To show that the condition (1) is sufficient, let us denote by $S_{i,j} = L_{k,j} \cap S_i$, $j \in \{1, \ldots, n\}, i \in \{1, \ldots, \text{maxflow}(t)\}$. It is sufficient to notice that $S_{i,j}$ are disjunctive partitions for every set $L_{k,j}$, i.e.,

$$\forall j \in \{1, \ldots, n\} : \bigcup_{i=1}^{\text{maxflow}(t)} S_{i,j} = L_{k,j}$$

and

$$\forall j_1, j_2 \in \{1, \ldots, n\} : S_{i,j_1} \cap S_{i,j_2} = \emptyset.$$

Since $|L_{k,j}| = k$, and the encoding of original n packets is done by M^{-1}, it follows that an eavesdropper can decode any original packet only by listening at least maxflow(t) links. ■

Example 3: We present an example that illustrates the security in our approach. The goal is to achieve secrecy[1] so that a passive adversary is able to reconstruct n source packets only when at least maxflow(t) links are eavesdropped. By sending XOR-ed packets on disjoint paths (exploiting the path diversity), an adversary is unable to decode the message although several paths are eavesdropped. Let us consider the network shown in Fig.3, where a source s communicates with two sinks t_1 and t_2 with the help of intermediate nodes u_i, $i = 1, 2, 3$, and sends twelve packets to t_1 and t_2. Packets are sent in four phases since maxflow(t) = 3. Let us use the following 5×12 Latin rectangle:

$$L_{5\times 12} = \begin{bmatrix} 4 & 2 & 11 & 8 & 12 & 1 & 9 & 5 & 10 & 7 & 6 & 3 \\ 2 & 8 & 12 & 3 & 6 & 10 & 4 & 11 & 5 & 1 & 9 & 7 \\ 3 & 4 & 2 & 9 & 11 & 12 & 5 & 6 & 7 & 8 & 10 & 1 \\ 9 & 10 & 1 & 6 & 3 & 7 & 2 & 8 & 4 & 11 & 12 & 5 \\ 6 & 5 & 7 & 10 & 1 & 11 & 8 & 3 & 12 & 4 & 2 & 9 \end{bmatrix}.$$

The colors of indexes in $L_{5\times 12}$ correspond to the colors of the packets as they are sent in Fig 3. If the sink nodes reconstruct the source packets with M^{-1}, then not all packets have the same level of decoding complexity. That is demonstrated with relations (2) and (3). For instance, to decode x_4, x_7 and x_8 nine coded packets are needed, while to decode x_5 and x_{11} just three packets are needed. The goal is to avoid this non-balanced complexity in the decoding. Therefore, as in Theorem 5 the encoding is done by M^{-1} and the decoding by M. When s computes the vector of coded packets as $\mathbf{c} = M^{-1}\mathbf{x}$, then the coded packets c_i, $i = 1, \ldots, 12$ are XOR-ed combinations of different number of source packets. Consequently, decoding of packets is done with a balanced matrix, i.e., $\mathbf{x} = M\mathbf{c}$.

[1]We use here the term *secrecy* as it is used in [13, Ch.7 pp. 185]

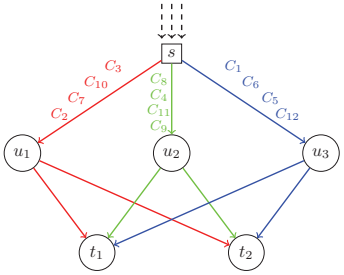


Figure 3: Routing of 12 packets for secure coding when decoding is performed with 5 coded packets

Assume that the routing is as follows: on the first path the source sends (C_3, C_{10}, C_7, C_2), on the second path (C_8, C_4, C_{11}, C_9) and (C_1, C_6, C_5, C_{12}) on the third path as it is shown in Fig.3. We use three different colors for the packets sent to three disjoint paths in order to demonstrate the essence of the proof of Theorem 5. Note that all colors are present in every column of the Latin rectangle $L_{5\times 12}$. This corresponds to the condition (1) in Theorem 5. In order to reconstruct at least one source packet, an adversary must eavesdrop at least 3 links.

$$
\begin{aligned}
c_1 &= x_2 \oplus x_3 \oplus x_4 \oplus x_6 \oplus x_9,\\
c_2 &= x_2 \oplus x_4 \oplus x_5 \oplus x_8 \oplus x_{10},\\
c_3 &= x_1 \oplus x_2 \oplus x_7 \oplus x_{11} \oplus x_{12},\\
c_4 &= x_3 \oplus x_6 \oplus x_8 \oplus x_9 \oplus x_{10},\\
c_5 &= x_1 \oplus x_3 \oplus x_6 \oplus x_{11} \oplus x_{12},\\
c_6 &= x_1 \oplus x_7 \oplus x_{10} \oplus x_{11} \oplus x_{12},\\
c_7 &= x_2 \oplus x_4 \oplus x_5 \oplus x_8 \oplus x_9,\\
c_8 &= x_3 \oplus x_5 \oplus x_6 \oplus x_8 \oplus x_{11},\\
c_9 &= x_4 \oplus x_5 \oplus x_7 \oplus x_{10} \oplus x_{12},\\
c_{10} &= x_1 \oplus x_4 \oplus x_7 \oplus x_8 \oplus x_{11},\\
c_{11} &= x_2 \oplus x_6 \oplus x_9 \oplus x_{10} \oplus x_{12},\\
c_{12} &= x_1 \oplus x_3 \oplus x_5 \oplus x_7 \oplus x_9.
\end{aligned}
\tag{2}
$$

$$
\begin{aligned}
x_1 &= c_2 \oplus c_5 \oplus c_{10} \oplus c_{11} \oplus c_{12},\\
x_2 &= c_1 \oplus c_3 \oplus c_5 \oplus c_6 \oplus c_7 \oplus c_9 \oplus c_{10},\\
x_3 &= c_3 \oplus c_5 \oplus c_6 \oplus c_7 \oplus c_8 \oplus c_9 \oplus c_{12},\\
x_4 &= c_4 \oplus c_5 \oplus c_6 \oplus c_7 \oplus c_8 \oplus c_9 \oplus c_{10} \oplus c_{11} \oplus c_{12},\\
x_5 &= c_1 \oplus c_2 \oplus c_4,\\
x_6 &= c_1 \oplus c_2 \oplus c_4 \oplus c_6 \oplus c_7 \oplus c_{10} \oplus c_{11},\\
x_7 &= c_2 \oplus c_3 \oplus c_4 \oplus c_5 \oplus c_6 \oplus c_7 \oplus c_8 \oplus c_{11} \oplus c_{12},\\
x_8 &= c_1 \oplus c_3 \oplus c_5 \oplus c_7 \oplus c_8 \oplus c_9 \oplus c_{10} \oplus c_{11} \oplus c_{12},\\
x_9 &= c_1 \oplus c_2 \oplus c_5 \oplus c_9 \oplus c_{10},\\
x_{10} &= c_1 \oplus c_5 \oplus c_7 \oplus c_9 \oplus c_{10},\\
x_{11} &= c_1 \oplus c_7 \oplus c_8,\\
x_{12} &= c_3 \oplus c_4 \oplus c_5 \oplus c_7 \oplus c_9.
\end{aligned}
\tag{3}
$$

V. Conclusions

In this paper we have presented a construction of codes over $GF(2)$ which reach the max-flow for single source multicast acyclic networks with delay. The coding is exclusively performed in $GF(2)$, i.e., it is a bitwise XOR of packets with equal lengths. The encoding and decoding are based on balanced nonsingular matrices that are obtained as incidence matrices from Latin rectangles. Balanced XOR-ed coding is of particular importance for energy and processor constraint devices. Additionally, we showed that the approach offers plausible security properties, i.e., if an eavesdropper wants to reconstruct at least one original packet, then the number of eavesdropped links must be equal to the max-flow of the network.

Possible future work includes intermediate nodes to form coded packets, as well as building networks dynamically by adding more and more sink nodes that reach the max-flow when the coding is XOR-ed coding.

Acknowledgements

We would like to thank Gergely Biczók for his discussions and remarks that significantly improved the paper.

REFERENCES

[1] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.

[2] K. Bhattad and K.R. Narayanan. Weakly secure network coding. *Proc. First Workshop on Network Coding, Theory, and Applications (NetCod)*, 2005.

[3] C. J. Colbourn and J. H. Dinitz. *Handbook of Combinatorial Designs, Second Edition (Discrete Mathematics and Its Applications)*. Chapman, Hall/CRC, 2006.

[4] C. J. Colbourn, J. H. Dinitz, and D. R. Stinson. Applications of combinatorial designs to communications, cryptography, and networking. 1999.

[5] P. Hall. On representatives of subsets. *J. London Math. Soc.*, 10(37):26–30, 1935.

[6] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong. A random linear network coding approach to multicast. *IEEE Transactions on Information Theory*, 52(10):4413–4430, 2006.

[7] M. T. Jacobson and P. Matthews. Generating uniformly distributed random latin squares. *Journal of Combinatorial Designs*, 4(6):405–437, 1996.

[8] S. Jaggi, Y. Cassuto, and M. Effros. Low complexity encoding for network codes. In *Information Theory, 2006 IEEE International Symposium on*, pages 40–44, 2006.

[9] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. XORs in the air: Practical wireless network coding. *IEEE/ACM Trans. Netw*, 16(3):497–510, 2008.

[10] R. Koetter and M. Médard. An algebraic approach to network coding. *IEEE/ACM Trans. Netw*, 11(5):782–795, 2003.

[11] E. Lawler. *Combinatorial Optimization : Networks and Matroids*. Dover Publications, 2001.

[12] S. Y. R. Li, R. W. Yeung, and N. Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49(2):371–381, 2003.

[13] M. Médard and A. Sprintson. *Network coding, Fundamentals and Applications*. 2012.

[14] M. V. Pedersen, F. H. P. Fitzek, and T. Larsen. Implementation and performance evaluation of network coding for cooperative mobile devices. In *Proc. IEEE Cognitive and Cooperative Wireless Networks Workshop*, 2008.

[15] J. Qureshi, Foh Chuan Heng, and Cai Jianfei. Optimal solution for the index coding problem using network coding over gf(2). In *Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 209–217, 2012.

[16] S. Riis. Linear versus nonlinear boolean functions in network flow. *CISS*, 2004.

[17] Salim Y. El Rouayheb, Alex Sprintson, and Costas N. Georghiades. On the index coding problem and its relation to network coding and matroid theory, September 30 2008. Comment: submitted to transactions on information theory.

[18] H. Shojania and B. Li. Random network coding on the iphone: fact or fiction? *NOSSDAV*, 2009.

[19] D. R. Stinson. *Combinatorial Designs: Constructions and Analysis*. SpringerVerlag, 2003.

[20] P. Vingelmann, M. V. Pedersen, F. H. P. Fitzek, and J. Heide. Multimedia distribution using network coding on the iphone platform. *Proceedings of the 2010 ACM multimedia workshop on Mobile cloud media computing*, 2010.

Paper 2

Families of Optimal Binary Non-MDS Erasure Codes

Danilo Gligoroski and Katina Kralevska

IEEE Proceedings on International Symposium on Information Theory (ISIT),
pp. 3150-3154, 2014

Families of Optimal Binary Non-MDS Erasure Codes

Danilo Gligoroski and Katina Kralevska
Department of Telematics; Faculty of Information Technology, Mathematics and Electrical Engineering; Norwegian University of Science and Technology, Trondheim, Norway,
Email: {danilog, katinak}@item.ntnu.no

Abstract

We introduce a definition for *Families of Optimal Binary Non-MDS Erasure Codes* for $[n, k]$ codes over $GF(2)$, and propose an algorithm for finding those families by using hill climbing techniques over Balanced XOR codes. Due to the hill climbing search, those families of codes have always better decoding probability than the codes generated in a typical Random Linear Network Coding scenario, i.e., random linear codes. We also show a surprising result that for small values of k, the decoding probability of our codes in $GF(2)$ is very close to the decoding probability of the codes obtained by Random Linear Network Coding but in the higher finite field $GF(4)$.

I. Introduction

In the fast approaching Zettabyte Era [3] the erasure codes will become the most important codes among all coding techniques. That is mostly due to two factors: 1. The global communications will be almost exclusively based on the packet switching paradigm, where the recovery from packet losses is addressed efficiently by erasure codes; 2. Storage systems will have capacities of hundreds of exabytes, and will have to tolerate and recover efficiently from multiple disk failures.

According to the rate of redundancy that is used, the erasure codes are divided in two classes: 1. Optimal or very close to optimal ones, known as Maximum Distance Separable (MDS) Codes [19], almost-MDS (AMDS) [5] and near-MDS codes (NMDS) [6], and 2: Suboptimal or non-MDS codes [7], [8], [11], [14], [17].

Reed-Solomon codes [22] are a well known class of MDS codes that provide a general technique for construction of MDS codes. However, these codes are defined in higher finite fields and they can be very computationally demanding. That is the main reason for series of research efforts to find codes that work just in the simplest finite field $GF(2)$ where the operations are bitwise exclusive-or (XOR) operations [2], [4], [12], [13].

Beside the use in massive storage systems, the erasure codes have been recently used in one research area that is addressing the demanding needs for increasing the speed and reliability of packet based communications. That evolving area is Network Coding [1]. Network Coding allows nodes in the network to perform a set of functions over the generated or received data packets before forwarding them. Random Linear Network Coding (RLNC) [10] is a network coding technique that produces random linear combinations of the packets over a Galois Field of size q, $GF(q)$. The field size has an impact on the decoding probability, i.e., the probability of receiving linearly independent packets increases with q.

When one or more sources want to transmit k packets to one or more destination nodes, the channel conditions must be considered. Even in a presence of packet losses (erasures) the destination node has to be able to decode k original packets by receiving $k + r$ packets. The authors in [18] derive the average number of received coded packets n for successful decoding of k original packets at the destination nodes. They study the effect of q on the performance of RLNC. The exact probability that k out of $k + r$ received packets are linearly independent is derived in [24]. Both papers show that q equal to 4 or 8 is enough to get very close to the optimal performance even when k is not very large.

However, as in the case of codes for massive storage systems, working in higher fields or with large number of data packets has an impact on the computational complexity leading to higher energy consumption

[9] and no real benefits. A recent result in [21] shows that the speed of computation on modern CPUs with wide SIMD instructions is similar for operations in $GF(2)$ and in $GF(16)$. On the other hand, implementing RLNC in higher fields on devices that have power and memory constraints is a challenging problem. Some recent studies show that RLNC in constrained devices in $GF(2)$ is up to two orders of magnitude less energy demanding and up to one order of magnitude faster than RLNC in higher fields [25], [20].

In this work we introduce a definition of *Families of Optimal Binary Non-MDS Erasure Codes* for $[n, k]$ codes over $GF(2)$. Then we propose one heuristic algorithm for finding those families by using hill climbing techniques over Balanced XOR codes introduced in [15]. Due to the hill climbing search, those families of codes have always better decoding probability than the codes generated in a typical Random Linear Network Coding scenario, i.e., random linear codes as described in [24]. We also show a surprising result that for small values of k, the decoding probability of our codes in $GF(2)$ is very close to the decoding probability of the codes obtained by RLNC but in the higher finite field $GF(4)$.

The paper is organized as follows. In Section II, we introduce the basic terminology and the definition of Families of Optimal Binary Non-MDS Erasure Codes. In Section III, we describe one heuristic algorithm for finding those Families of Optimal Binary non-MDS Erasure Codes. We also discuss and compare the properties of our erasure codes to codes generated in a typical Random Linear Network Coding scenario, i.e., random linear codes. Conclusions and future work are summarized in Section IV.

II. Mathematical Preliminaries

In this section we briefly introduce the basic terminology, some useful properties and facts about linear codes, as well as some basic terminology and coding methods for Balanced XOR codes [15].

Let us denote by $\mathbf{F}_q = GF(q)$ the Galois field with q elements, and by $\mathbf{F}_q^n$ the n-dimensional vector space over $\mathbf{F}_q$. Let us also denote by $[n, k]_q$ the q-ary linear code of length n and rank k which is actually a linear subspace C with dimension k of the vector space $\mathbf{F}_q$. An $[n, k, d]_q$ code is an $[n, k]_q$ code with minimum weight at least d among all nonzero codewords. An $[n, k, d]_q$ code is called maximum distance separable (MDS) if $d = n - k + 1$. The Singleton defect of an $[n, k, d]_q$ code C defined as $s(C) = n - k + 1 - d$ measures how far away is C from being MDS.

Below we give some basic properties for MDS matrices that we use in this paper:

Proposition 1 ([19], Ch. 11, Corollary 3): Let C be an $[n, k, d]$ code over $GF(q)$. The following statements are equivalent:

1) C is MDS;
2) every k columns of a generator matrix G are linearly independent;
3) every $n - k$ columns of a parity check matrix H are linearly independent.

Definition 1: Let C be an $[n, k]$ code over $GF(q)$ with a generator matrix G. Let us denote by $\mathcal{G}_I, I = k, \ldots, n$ the sets of submatrices obtained from G when choosing I columns from G, and by $\mathcal{D}_I \subset \mathcal{G}_I, I = k, \ldots, n$ the subsets of $\mathcal{G}_I$ with a rank k. We call the following vector $V_D = (\varrho_0, \varrho_1, \ldots, \varrho_{n-k})$, $\varrho_i = |\mathcal{D}_{i+k}|/|\mathcal{G}_{i+k}|$, the *Vector of Exact Decoding Probability*, for the code C.

With other words, the value ϱ_i represents the probability that we can decode all k original values $x_1, \ldots, x_k$, if we are given $k + i$ values $y_1, \ldots, y_{k+i}$ that corresponds to encoding with $k + i$ columns of the generator matrix G.

For random generator matrices G, the values of V_D are calculated in [24] and we formulate them in the following Proposition:

Proposition 2: For a linear $[n, k]$ code over $GF(q)$ with a random generator matrix G the elements of the vector $V_D = (\varrho_0, \varrho_1, \ldots, \varrho_{n-k})$ have the following values:

$$\varrho_i = P(k + i), \tag{1}$$

where the values $P(I)$ are computed as follows:

$$P(I) = \begin{cases} 0 & \text{if } I < k, \\ \prod_{j=0}^{k-1} \left(1 - \frac{1}{q^{I-j}}\right) & \text{if } I \geq k. \end{cases} \tag{2}$$

Proof: The equation (2) is actually the equation (7) in [24] with adopted notation to be consistent with the standard notation for linear $[n, k]$ codes over $GF(q)$. The equation (1) then follows directly. ■

The connection between the Vector of Exact Decoding Probability and the MDS codes can be established by using the Proposition 2 as follows:

Theorem 1: A linear $[n, k]$ code C over $GF(q)$ with a generator matrix G is a MDS code iff the Vector of Exact Decoding Probability is the following vector $V_D = (\varrho_0, \varrho_1, \ldots, \varrho_{n-k}) = (1, 1, \ldots, 1)$.

Proof: The theorem can be proved with a direct application of the Proposition 2 and the Definition 1. ■

In this work we are interested exclusively to work with XOR coding, i.e., to work with linear binary codes. Thus, our interest is to define a class of binary codes that in some properties are as close as possible to MDS codes. Unfortunately, it is a well known old fact in coding theory (see for example [19]) that for the case of linear binary codes, all MDS codes are trivial, i.e., $k = 1$ or $n = k + 1$ or $n = k$.

So, dealing with the fact that non-trivial binary codes are not MDS, we adopt a strategy to search for codes that will be optimal from certain perspective according to the Vector of Exact Decoding Probability V_D. When a channel has an erasure probability p the strategy will be to find binary codes that maximize the probability to recover the original data. Therefore, we prove the following Theorem:

Theorem 2: Let C be a binary linear $[n, k]$ code with a Vector of Exact Decoding Probability $V_D = (\varrho_0, \varrho_1, \ldots, \varrho_{n-k})$ and let k packets are encoded by C. The probability p_s of successful decoding of k packets from n encoded and transmitted packets via a channel with an erasure probability p is:

$$p_s = 1 - \left(\sum_{i=0}^{n-k} \binom{n}{i} p^i (1-p)^{n-i} (1 - \varrho_{n-k-i}) + \sum_{i=n-k+1}^{n} \binom{n}{i} p^i (1-p)^{n-i} \right) \tag{3}$$

Proof: Let us denote by E_1 the event that i packets, where $0 \leq i \leq n-k$, are lost during the transmission, and by E_2 the event that more than $n-k$ packets from the set of all n packets are lost during the transmission.

The probability of the event E_1 is calculated by the expression:

$$P(E_1) = \sum_{i=0}^{n-k} \binom{n}{i} p^i (1-p)^{n-i}, \tag{4}$$

and the probability of the event E_2 is:

$$P(E_2) = \sum_{i=n-k+1}^{n} \binom{n}{i} p^i (1-p)^{n-i}. \tag{5}$$

From expression (4) we compute the probability p_{u_1} of failure to decode k original packets, by multiplying every value in the sum by the opposite probability of successful decoding when $n-k-i$ columns of the generator matrix G are received, i.e., when i packets are lost. So the decoding failure probability if i packets are lost ($0 \leq i \leq n-k$) is computed by the following expression:

$$p_{u_1} = \sum_{i=0}^{n-k} \binom{n}{i} p^i (1-p)^{n-i} (1 - \varrho_{n-k-i}). \tag{6}$$

If more than $n-k$ packets are lost then the probability to fail the decoding is 100% thus the probability p_{u_2} of failure to decode k original packets is equal to $P(E_2)$, i.e., $p_{u_2} = P(E_2)$.

TABLE I: A general Stochastic Hill-Climbing algorithm for finding a Family of Optimal Binary Non-MDS Erasure Codes for given values of n and k

Algorithm 1
Input. n and k
Output. A candidate Family $\mathcal{C}$ of Optimal Binary Non-MDS Erasure Codes
1. Find a random $[n,k]$ linear binary code and compute its Vector of Exact Decoding Probability $V_D = (\varrho_0, \varrho_1, \ldots, \varrho_{n-k})$ and its probability p_s of successful decoding of k packets from the equation (3). 2. Repeatedly improve the solution until no more improvements are necessary/possible.

In total, the probability of unsuccessful decoding p_u is:

$$\begin{aligned} p_u &= p_{u_1} + p_{u_2} = \\ &= \sum_{i=0}^{n-k} \binom{n}{i} p^i (1-p)^{n-i} (1 - \varrho_{n-k-i}) + \\ &\quad + \sum_{i=n-k+1}^{n} \binom{n}{i} p^i (1-p)^{n-i} \end{aligned} \tag{7}$$

Finally the probability p_s of successful decoding of k packets is the opposite probability of p_u i.e.,

$$p_s = 1 - p_u.$$

■

Having defined the probability p_s of successful decoding of k packets that are encoded with an $[n,k]$ binary code, we define a *Family of Optimal Binary Non-MDS Erasure Codes* as follows:

Definition 2: Let $\mathcal{C}$ be a family of binary linear $[n,k]$ codes that have a probability p_s of successful decoding k packets from n encoded and transmitted packets via a channel with an erasure probability p. We say that $\mathcal{C}$ is a *Family of Optimal Binary Non-MDS Erasure Codes* if for every binary linear $[n,k]$ code C' with a probability p'_s of successful decoding of k packets in a channel with an erasure probability p, there exist a code $C \in \mathcal{C}$ with a probability p_s of successful decoding, such that $p'_s \leq p_s$, for every erasure probability p.

Problem 1: For given values of n and k find a Family $\mathcal{C}$ of Optimal Binary Non-MDS Erasure Codes.

III. A Hill Climbing Heuristics For Finding Families of Optimal Binary Non-MDS Erasure Codes

Finding exact analytical solution (or finding deterministic and efficient algorithm that will find the solution) for the Problem 1 is hard and in this moment we do not know such a solution. However, there are many heuristic optimization methodologies that can be used for a search of approximate solutions. We choose to use the simplest one: The Stochastic Hill-Climbing Methodology [23]. The hill climbing heuristics has been already used in optimizing problems for RLNC such as in [16]. In general, the stochastic heuristics is defined as in Algorithm 1.

In order to improve the codes found by Algorithm 1 we decided to work with balanced structures as they were introduced in [15].

Definition 3: A XOR-ed coding is a coding that is realized exclusively by bitwise XOR operations between packets with equal length. Hence, it is a parallel bitwise linear transformation of k source bits $x = (x_1, \ldots, x_k)$ by a $k \times k$ nonsingular binary matrix $\mathbf{K}$, i.e., $y = x \cdot \mathbf{K}$.

In other words XOR-ed coding assumes work within the smallest finite field $GF(2)$, i.e., with $k \times k$ nonsingular binary matrices $\mathbf{K}$. While the binary matrices $\mathbf{K}$ in general can be of any form, the specifics about matrices introduced in [15] are that they are highly structured, balanced and their construction is based on Latin rectangles of dimensions $k_1 \times k$.

Definition 4: A Latin square of order k with entries from an k-set X is an $k \times k$ array L in which every cell contains an element of X such that every row of L is a permutation of X and every column of L is a permutation of X.

Definition 5: A $k_1 \times k$ Latin rectangle is a $k_1 \times k$ array (where $k_1 \leq k$) in which each cell contains a single symbol from an k-set X, such that each symbol occurs exactly once in each row and at most once in each column.

Definition 6: Let (X, A) be a design where $X = \{x_1, \ldots, x_v\}$ and $A = \{A_1, \ldots, A_b\}$. The incidence matrix of (X, A) is the $v \times b$ 0-1 matrix $M = (m_{i,j})$ defined by the rule $m_{i,j} = \begin{cases} 1, & \text{if } x_i \in A_j, \\ 0, & \text{if } x_i \notin A_j. \end{cases}$

Proposition 3 ([15]): The incidence matrix $M = (m_{i,j})$ of any Latin rectangle with dimensions $k_1 \times k$ is a balanced matrix with k_1 ones in each row and each column.

Proposition 4 ([15]): The necessary condition an incidence matrix $M = (m_{i,j})$ of a $k_1 \times k$ Latin rectangle to be nonsingular in $GF(2)$ is k_1 to be odd, i.e., $k_1 = 2l + 1$.

Example 1: Let us take the following Latin square and split it into two Latin rectangles:

$$L = \left[\begin{array}{ccccc} 1 & 4 & 3 & 5 & 2 \\ 3 & 1 & 5 & 2 & 4 \\ 4 & 2 & 1 & 3 & 5 \\ \hline 5 & 3 & 2 & 4 & 1 \\ 2 & 5 & 4 & 1 & 3 \end{array}\right].$$

The incidence matrix M of the 3×5 upper Latin rectangle is:

$$M = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Note how balanced are the rows and columns: in every row and every column, the number of 1s is 3.

The following proposition follows directly from the Proposition 4:

Proposition 5: The $k+1$-th column of the generator matrix G of a trivial $[k+1, k]_2$ MDS code that has in the first k columns a matrix for a balanced XOR-ed coding consists of all 1s.

We now describe the modified Stochastic Hill-Climbing that is using Balanced XOR codes where one column of the generator matrix is defined as in Proposition 5:

We would like to note that Algorithm 1 can find codes with similar decoding probabilities as Algorithm 2, but after performing more stochastic search attempts. Moreover, the codes that Algorithm 2 finds have advantages that they are structured, balanced and they are sparse, where the sparsity can go down to just 3 nonzero positions.

We now give two numerical results that compare the performance of our codes to a typical linear random code in $GF(2)$ that can be generated in RLNC. The same parameters are taken as in [24], i.e., $r = 0, \ldots, 8$ is the number of excess packets for $k = 5$ and $k = 100$. The results show that the decoding probability with our scheme is closer to the decoding probability under RLNC in $GF(4)$ when k is small. We would like to emphasize that with Algorithm 2 we could easily find codes with k in range $[5, \ldots, 1000]$.

In Figure 1 the code that was found after 10,000 stochastic attempts by the Balanced XOR-ed approach of Algorithm 2 is based on the Latin Square from Example 1. Its generator matrix is the following:

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

TABLE II: A Stochastic Hill-Climbing algorithm for finding a Family of Optimal Binary Non-MDS Erasure Codes based on Balanced XOR codes

Algorithm 2
Input. n and k
Output. A candidate Family $\mathcal{C}$ of Optimal Binary Non-MDS Erasure Codes
1. Find a random Balanced XOR code and put it as the first part of the generator matrix G of an $[n, k]$ code. Set the $k+1$-th column to consists of all 1s, and set the remaining columns with random values. Compute the Vector of Exact Decoding Probability $V_D = (\varrho_0, \varrho_1, \ldots, \varrho_{n-k})$ and its probability p_s of successful decoding of k packets from the equation (3). 2. Repeatedly improve the solution until no more improvements are necessary/possible.

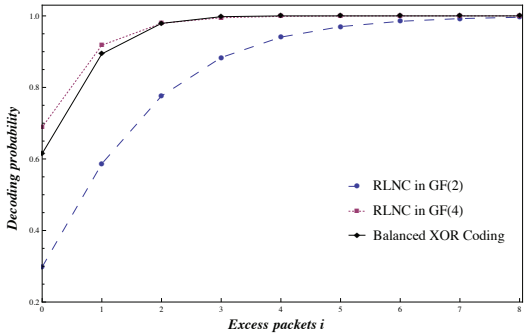


Figure 1: Vector of Exact Decoding Probability V_D for k=5

The Vector of Exact Decoding Probability for this code is: $V_D = (0.615, 0.895, 0.979, 0.998, 1., 1., 1., 1.)$ and is presented in Figure 1 with a solid line.

A typical random linear code in $GF(2)$ generated in RLNC is presented in Figure 1 with a dashed line. For comparison purposes, we put the values for decoding probabilities of a typical random linear code in $GF(4)$ in the same Figure 1. As it can be seen, our codes in $GF(2)$ have decoding probabilities as a random linear code in $GF(4)$.

The real advantage of our codes is seen in Figure 2 in channels where packet losses occur with certain probabilities. Similarly as in [24] we give the results for $[n, k] = [108, 100]$ in Figure 3 and in Figure 4.

IV. Conclusions

We introduced a definition of *Families of Optimal Binary Non-MDS Erasure Codes* for $[n, k]$ codes over $GF(2)$ and proposed one heuristic algorithm for finding those families using hill climbing techniques over Balanced XOR codes. We showed that the families of codes that we found have always better decoding probability than the decoding probability of random linear codes generated in RLNC. We also showed that for small values of k the decoding probability of our codes in $GF(2)$ is very close to the decoding probability of the random linear codes in $GF(4)$.

As a next research direction, we point out that it will be very useful to further investigate the theoretical lower and upper bounds of decoding probabilities of the defined Families of Optimal Binary Non-MDS Erasure Codes and to find better heuristic or deterministic algorithms for efficient finding of those families. It would be a natural research directions to see how this methodology performs in higher fields.

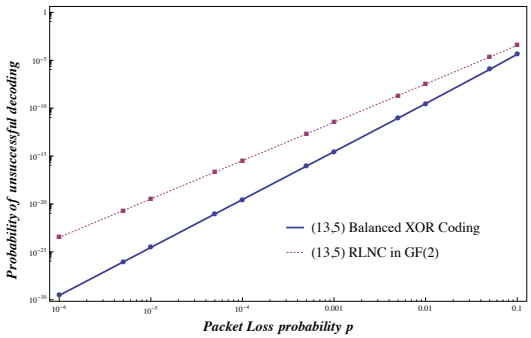


Figure 2: Comparison between probabilities of unsuccessful decoding of a typical RLNC code and a code obtained with our stochastic strategy in $GF(2)$ for $k = 5$

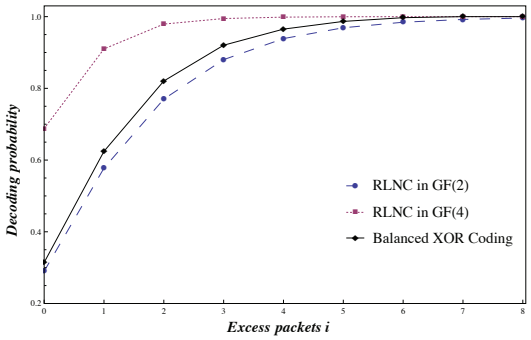


Figure 3: Vector of Exact Decoding Probability V_D for k=100

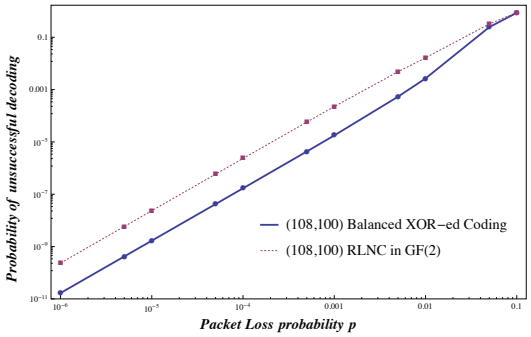


Figure 4: Comparison between probabilities of unsuccessful decoding of a typical RLNC code and a code obtained with our stochastic strategy in $GF(2)$ for $k = 100$

ACKNOWLEDGEMENTS

We would like to thank Harald Øverby and Rune E. Jensen for their discussions that improved the quality of this paper. We would also like to thank the anonymous reviewers for their useful comments and suggestions.

REFERENCES

[1] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.

[2] Mario Blaum, Jim Brady, Jehoshua Bruck, and Jai Menon. Evenodd: An efficient scheme for tolerating double disk failures in raid architectures. *IEEE Trans. Computers*, 44(2):192–202, 1995.

[3] Cisco. Cisco visual networking index: Forecast and methodology, 20122017. *White Paper*, May 2013.

[4] Peter F. Corbett, Robert English, Atul Goel, Tomislav Grcanac, Steven Kleiman, James Leong, and Sunitha Sankar. Row-diagonal parity for double disk failure correction. In *FAST*, pages 1–14. USENIX, 2004.

[5] Mario A. de Boer. Almost mds codes. *Des. Codes Cryptography*, 9(2):143–155, 1996.

[6] S.M. Dodunekov and I.N. Landjev. On near-mds codes. *Journal of Geometry*, 54:30–43, 1995.

[7] Kevin M. Greenan, Xiaozhou Li, and Jay J. Wylie. Flat xor-based erasure codes in storage systems: Constructions, efficient recovery, and tradeoffs. In *MSST*, pages 1–14. IEEE Computer Society, 2010.

[8] James Lee Hafner. Weaver codes: Highly fault tolerant erasure codes for storage systems. In *FAST*. USENIX, 2005.

[9] Janus Heide, Morten V. Pedersen, Frank H. P. Fitzek, and Muriel Médard. On code parameters and coding vector representation for practical RLNC. In *ICC*, pages 1–5. IEEE, 2011.

[10] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong. A random linear network coding approach to multicast. *IEEE Transactions on Information Theory*, 52(10):4413–4430, 2006.

[11] Cheng Huang, Minghua Chen, and Jin Li. Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems. *TOS*, 9(1):3, 2013.

[12] Cheng Huang and Lihao Xu. Star : An efficient coding scheme for correcting triple storage node failures. *IEEE Trans. Computers*, 57(7):889–901, 2008.

[13] O. Khan, R. Burns, J. S. Plank, W. Pierce, and C. Huang. Rethinking erasure codes for cloud file systems: Minimizing I/O for recovery and degraded reads. In *FAST-2012: 10th Usenix Conference on File and Storage Technologies*, San Jose, February 2012.

[14] A. Kiani and S. Akhlaghi. A non-mds erasure code scheme for storage applications. *Journal of Communication Engineering*, 2.

[15] K. Kralevska, D. Gligoroski, and H. Øverby. Balanced XOR-ed coding. In *Advances in Communication Networking - 19th EUNICE/IFIP*, volume 8115 of *LNCS*, pages 161–172. Springer, 2013.

[16] E. Kurdoglu, N. Thomos, and P. Frossard. Scalable video dissemination with prioritized network coding. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pages 1–6, 2011.

[17] Michael Luby, Michael Mitzenmacher, Mohammad Amin Shokrollahi, Daniel A. Spielman, and Volker Stemann. Practical loss-resilient codes. In *STOC*, pages 150–159. ACM, 1997.

[18] Daniel Enrique Lucani, Muriel Médard, and Milica Stojanovic. Random linear network coding for time-division duplexing: Field size considerations. In *GLOBECOM*, pages 1–6. IEEE, 2009.

[19] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-holland Publishing Company, 2nd edition, 1978.

[20] M. V. Pedersen, J. Heide, F.H.P. Fitzek, and T. Larsen. Network coding for mobile devices - systematic binary random rateless codes. In *Workshop on Cooperative Mobile Networks 2009 - ICC09*. IEEE, June 2009.

[21] J. S. Plank, K. M. Greenan, and E. L. Miller. Screaming fast Galois Field arithmetic using Intel SIMD instructions. In *FAST-2013: 11th Usenix Conference on File and Storage Technologies*, San Jose, February 2013.

[22] Irving Reed and Golomb Solomon. Polynomial codes over certain finite fields. *Journal of the Society of Industrial and Applied Mathematics*, 8(2):300–304, 06/1960 1960.

[23] S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education Inc., 2003.

[24] Oscar Trullols-Cruces, Jose Maria Barcelo-Ordinas, and Marco Fiore. Exact decoding probability under random linear network coding. 2011.

[25] P. Vingelmann, M. V. Pedersen, F. H. P. Fitzek, and J. Heide. Multimedia distribution using network coding on the iphone platform. *Proceedings of the 2010 ACM multimedia workshop on Mobile cloud media computing*, 2010.

Paper 3

Minimal Header Overhead for Random Linear Network Coding

Danilo Gligoroski, Katina Kralevska, and Harald Øverby

IEEE International Conference on Communication Workshop (ICCW), pp. 680-685, 2015

Minimal Header Overhead for Random Linear Network Coding

Danilo Gligoroski, Katina Kralevska, and Harald Øverby
Department of Telematics; Faculty of Information Technology, Mathematics and Electrical Engineering; Norwegian University of Science and Technology, Trondheim, Norway,
Email: danilog@item.ntnu.no, katinak@item.ntnu.no, haraldov@item.ntnu.no

Abstract

The energy used to transmit a single bit of data between the devices in wireless networks is equal to the energy for performing hundreds of instructions in those devices. Thus the reduction of the data necessary to transmit, while keeping the same functionality of the employed algorithms is a formidable and challenging scientific task. We describe an algorithm called Small Set of Allowed Coefficients (SSAC) that produces the shortest header overhead in random linear network coding schemes compared with all other approaches reported in the literature. The header overhead length is 2 to 7 times shorter than the length achieved by related compression techniques. For example, SSAC algorithm compresses the length of the header overhead in a generation of 128 packets to 24 bits, while the closest best result achieved by an algorithm based on error correcting codes has a header overhead length of 84 bits in $GF(16)$ and 224 bits in $GF(256)$. We show that the header length in SSAC does not depend on the size of the finite field where the operations are performed, i.e., it just depends on the number of combined packets m.

Keywords: Network coding, Header overhead, Compressed header overhead

I. Introduction

The main feature of network coding is enabling the intermediate nodes in a multi-hop network to perform coding [1]. All nodes in the network excluding the sink nodes perform random linear mappings (RLNC [8]) from input packets into output packets over a Galois Field of size q, $GF(q)$. The coding operations that are done over a packet are recorded in the packet header as a vector of coefficients. In the multi-hop networks, the vector of coefficients is updated at each node that performs network coding. The sink nodes decode the data based on the coefficients in the packet header.

One of the main challenges of implementing network coding is the header overhead imposed by the coding coefficients. When a source wants to send a large file, the file is split into several generations each consisting of n packets. The length of the vector of coefficients is $n \log_2 q$ bits under RLNC in the finite Galois Field $GF(q)$. As the number of packets in a generation or the size of the Galois Field increases, the length of the header overhead due to the coding coefficients becomes significant. This affects the goodput of the system and can be a significant contribution to the system load for some network scenarios.

Additionally, it is a known scientific fact that the energy used to transmit a single bit of data between the devices in ad hoc sensor networks is equal to the energy for performing 800 instructions in the devices [12]. Thus the reduction of the data necessary to transmit, while keeping the same functionality of the employed algorithms is a formidable and challenging scientific task that implies that many applications will benefit by performing local computations rather than sending more bits.

In this paper, we present a novel approach for practical network coding called Small Set of Allowed Coefficients (SSAC). The main contribution of this approach is that it generates the shortest compressed network coding header compared to related approaches reported in the literature. In SSAC the header length does not depend on the size of the finite field where the operations are performed, but only on the number of combined packets m.

The paper is organized as follows: Related work is presented in Section II. In Section III, we present the Small Set of Allowed Coefficients algorithm. Experimental results are reported in Section IV. Conclusions and future work are summarized in Section V.

II. Related Work

Several papers in recent literature have addressed the problem of reducing the header overhead in network coding.

The first suggested solution was done in [9]. This approach finds a smaller vector subspace of the original vector space, and the coding is done just in that vector subspace. By this method, finding a proper subspace can be a computational challenge and decoding at a sink node is also a challenging task since every combination of source data should result in a distinct union subspace.

The concept of sparse coding is well known and it was first proposed in [14] for header compression in network coding, to reduce the number of combined packets in one coded packet from n to m where $m < n$. This scheme uses parity-check matrices of error correcting codes to compress the header length down to $\mathcal{O}(m \log_2 n \log_2 q)$ bits. As noted in [14], the number of sources in sensor networks is large and a typical frame length is 30 bytes for data transmission. Consider a sensor network where 60 nodes send data. If RLNC in $GF(16)$ is performed, then 30 bytes per packet are used for recording the coding coefficients, i.e., the length of the header overhead is equal to the length of the useful data. Therefore, the authors of [14] introduce the idea of compressing the coding vectors. The length of the coding vector is reduced by limiting the number of packets that are combined in a coded packet denoted by m. However, limiting the number of packets being combined affects the invertibility of the matrix or decreases the probability of a redundant packet being innovative [4], [6], [7], [13].

The authors in [10] proposed improved schemes for compression of the coding vectors by using erasure decoding and list decoding. The compressed header length under the erasure decoding scheme is $m + n/\log_2 q$. The header length becomes arbitrarily close to $m + O(\log_2 n)/\log_2 q$ when the list decoding scheme is used. The both schemes are valid for moderate or large value of m.

Another completely different approach with a fixed and small header overhead was proposed in [5]. There, the header overhead is the seed for generating the coding coefficients with a known pseudo-random number generator (PRNG). This effectively reduces the header overhead to the size of the seed. However, as noted in [11] this approach does not support re-encoding which is the crucial constituent of the random linear network coding.

A similar approach from the point of view of the extremity of reducing the overhead just to one symbol is proposed in [15]. There, the generation of the coding coefficients is based on modified Vandermonde matrices which can be determined by one symbol. However, the two big constraints of this design are: the network coding nodes should only perform addition operations and the generation size is upper bounded by $\log_2 q$ due to the cyclic property of the matrices.

We evaluate the presented approaches by using two metrics: the header length and the number of packets combined in a coded packet. An overview is given in Table I. The features of SSAC are also presented in Table I and we discuss them in the next Sections. Some of the presented methods do not support re-encoding or are valid for restricted set of m. Therefore, we compare SSAC with traditional RLNC and error correcting codes in Section IV.

III. The Algorithm: *Small Set of Allowed Coefficients (SSAC)*

We denote by $GF(q)$ a finite field (Galois Field) with q elements where q is power of 2. It is known that for any finite field $GF(q)$ the set of all nonzero elements $GF(q)^\times = GF(q) \setminus 0$ form a multiplicative cyclic group $(GF(q)^\times, \times)$. That means that any nonzero element $\beta \in GF(q)^\times$ can be represented as a power of a single element $\alpha \in GF(q)^\times$, i.e., $\beta = \alpha^r$ for some $r \leq q$. Such a generator α is called a *primitive element* of the finite field.

TABLE I: Comparison of header length in bits for different network coding schemes when the generation size is n

Scheme	Header length	Packets combined m	Operations in		
			Sources	Intemediate nodes	Destinations
RLNC	$n \log_2 q$	n	$GF(q)$	$GF(q)$	Gaussian elimination
Error correcting codes [14]	$\mathcal{O}(m \log_2 n \log_2 q)$	$\log_2 n < m \leq \lfloor (n-k)/2 \rfloor$	$GF(q)$	$GF(q)$	Berlekamp-Massey
Seed with PRNG [5]	Size of the seed	n	$GF(q)$	Do not support	Gaussian elimination
Erasure decoding [10]	$m \log_2 q + n$	Moderate or large m	$GF(q)$	$GF(q)$	Berlekamp-Massey
List decoding [10]	$m \log_2 q + \mathcal{O}(\log_2 n)$	Moderate or large m	$GF(q)$, q is large	$GF(q)$, q is large	Berlekamp-Massey
Vandermonde matrices [15]	$\log_2 q$	$m \leq \log_2 q$	$GF(q)$	$GF(2)$	Gaussian elimination
SSAC	$m(\log_2 \lvert Q \rvert + \log_2 n)$	m	$GF(q)$	$GF(q)$	Gaussian elimination

We consider that one or several sources send n original data packets through a network where the source(s) and intermediate nodes can perform random linear network coding. We describe an algorithm which aims to provide a minimal header overhead for random linear network coding. The algorithm is based on utilizing a small set $Q \subset GF(q)$ of coefficients that multiply the original data. We formalize this with the following definition:

Definition 1: For a subset Q of $GF(q)$ we say that it is a *Small Set of Allowed Coefficients (SSAC)* if all operations of multiplication of the original data packets in the network coding procedures are performed *only* by the elements of Q.

Note that due to trivial reasons of impossible representation of the packet transformations, the set Q cannot have just one element.

The relation between the compression techniques presented in [14] and [10] and our approach can be described as follows: for the set Q in [14] and [10] they use all non-zero elements from the finite field $GF(q)$, while we use much smaller set. Namely, we use only two elements, i.e., $Q = \{q_0, q_1\}$ where both elements q_0 and q_1 are primitive elements in $GF(q)$.

Another crucial part of our method is the initial sparse encoding of the original data. Let us denote by $\mathbf{x} = (x_1, \ldots, x_n)$ a generation of n original data packets. We set the level of sparsity to be a small number m, $m = 2, 3$ or 4 as reported in [14]. In the beginning, the source is generating a $k \times n$, $(k \geq n)$ random sparse matrix

$$\mathbf{E}_{k \times n} = \begin{pmatrix} \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_k \end{pmatrix} \tag{1}$$

where every row-vector $\mathbf{e}_i$ is a sparse n-dimensional vector with just m non-zero elements from the set Q. It uses $\mathbf{E}_{k \times n}$ to encode the initial data packets. However, in that encoding, due to the sparsity of the rows, instead of putting the whole rows as header overheads for each of the packets, it uses a special compression format.

Definition 2: We say that the row-vector $\mathbf{e}$ is encoded in *Compressed Sparse Row (CSR)* format if it is presented in the form: $\mathbf{h} = (i_1 || j_1 || \ldots || i_m || j_m)$ where i_μ denotes the index of an element of the set Q that is in the row vector $\mathbf{e}$ at the j_μ position, where j_μ is in binary format.

Compressed Sparse Row (CSR) is applied frequently in mathematics and computing, and here we refer an interested reader to [3] as a good starting reference. Note that there is a slight difference between the described compressed formats in [3] and our format, since we apply the compressed sparse coding for each row separately (not for the whole matrix) due to the nature of the network coding paradigm where packets are transmitted through the network together with their header overhead. Thus we adopt the following convention:

Definition 3: All packets transmitted in the network have the format $\mathbf{h}||P$. The value of $\mathbf{h}$ is the header overhead as defined in Definition 2, where the indexes j_μ denote the indexes of original data packets that are multiplied by the corresponding element $i_\mu \in Q$. The value of P is the data payload for that packet.

Without a proof we give here the following proposition:

Proposition 1: The length of $\mathbf{h}$ is $m(\log_2 |Q| + \log_2 n)$ bits.

It follows immediately that for a fixed value of n, the size of the header overhead $\mathbf{h}$ in bits is minimal if the size of the set Q is minimal.

Proposition 2: If the set Q has two elements, then for any number of original packets n, the size of the header overhead $\mathbf{h}$ encoded as in Definition 3 achieves the minimum value of $m(1+\log_2 n)$ bits.

We give here a complete small example for coding of $n=8$ packets in $GF(16)$.

Example 1: Let us use the following irreducible polynomial $i(x)=x^4+x^3+1$ in the finite field $GF(2^4)=GF(16)$. Let us choose the following two primitive elements: $q_0=\mathbf{4}\equiv(0,1,0,0)_{16}\equiv 0x^3+1x^2+0x+0$ and $q_1=\mathbf{14}\equiv(1,1,1,0)_{16}\equiv 1x^3+1x^2+1x+0$. Note that we denote the elements of the field with bold integers to distinguish them from ordinary integers. Note also that the integer binary representation in four bits corresponds with their polynomial representation with coefficients $\{0,1\}$ and with polynomials of degree up to 3.

Let us consider that $n=8$ packets and $m=3$. If we have the following vector: $\mathbf{w}=(\mathbf{0},\mathbf{4},\mathbf{4},\mathbf{0},\mathbf{14},\mathbf{0},\mathbf{0},\mathbf{0})$, then it is represented in the following CSR format: $\mathbf{h}=(0\,001\,0\,010\,1\,100)$. Note that the spacing is just for readability, and that the indexing of the n coordinates is done in the range from 0 to $n-1$.

Let us now suppose that a node has received 5 random linear network coded packets, encoded with the following sparse row vectors:

$$\mathbf{E}_{5\times 8}=\begin{pmatrix}\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{4}&\mathbf{0}&\mathbf{0}&\mathbf{14}&\mathbf{4}\\ \mathbf{4}&\mathbf{0}&\mathbf{4}&\mathbf{0}&\mathbf{0}&\mathbf{14}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{14}&\mathbf{0}&\mathbf{14}&\mathbf{4}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\mathbf{4}&\mathbf{0}&\mathbf{14}&\mathbf{0}&\mathbf{14}&\mathbf{0}\\ \mathbf{0}&\mathbf{14}&\mathbf{0}&\mathbf{14}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{14}\end{pmatrix}$$

Note that $\mathbf{E}_{5\times 8}$ is similar to (1), but since it is a node (not the source) the number of rows $k=5$ is less than n. We say that number k in every node represents *the number of buffered packets in that node*. If the node combines all of its buffered packets (in this case 5 packets) it may find a new innovative packet which is a combination of the original data packets, with a vector that has only m nonzero elements from the set Q. Indeed, if the buffered packets are combined with the following vector: $\mathbf{x}=(\mathbf{1},\mathbf{0},\mathbf{0},\mathbf{1},\mathbf{5})$ then the new innovative packet is $\mathbf{e}_6=\mathbf{w}=\mathbf{x}\cdot\mathbf{E}_{5\times 8}=(\mathbf{0},\mathbf{4},\mathbf{4},\mathbf{0},\mathbf{14},\mathbf{0},\mathbf{0},\mathbf{0})$. By *innovative* we mean that it is linearly independent from all existing rows in the matrix $\mathbf{E}_{5\times 8}$. Then the node generates the compressed header $\mathbf{h}_6=(0\,001\,0\,010\,1\,100)$ and with the vector $\mathbf{x}$ encodes the buffered 5 packets producing the innovative data payload P. It sends the packet $\mathbf{h}_6||P$ as presented in Definition 3 .

We systematize the previous example in a form of a precise step by step algorithm SSAC in Table II.

A. Efficiency of SSAC

The procedure called RandomSparseVector(m,n,Q) in Step 4 returns one random n dimensional vector that is sparse and has exactly m nonzero coordinates from the set Q. It is important to notice that the algorithm is a probabilistic one, and in Step 5 it attempts to find a solution $\mathbf{x}$ of a linear matrix-vector equation

$$\mathbf{w}=\mathbf{x}\cdot\mathbf{E}. \tag{2}$$

Depending on the values of k, m, n and $\mathbf{E}$ it is not always possible to find such a linear solution $\mathbf{x}$. One work that addresses the problem of existence of solutions of equation (2) is [2]. We summarize the findings in [2] adopted for our SSAC algorithm with the following Observation:

Observation 1: Let $\mathbf{E}_{k\times n}$ be a random sparse matrix where the sparsity of each row is such that there are exactly m nonzero elements from the set of coefficients Q. If the number of rows k is

$$k=k_{opt}\approx m\log\frac{n}{m}, \tag{3}$$

then there exists a sparse vector $\mathbf{w}$ (with a sparsity of having exactly m nonzero elements from the set of coefficients Q) and a solution to the equation (2) with a probability $1-\epsilon$, where ϵ is a small value.

TABLE II: An algorithm for network coding that generates an innovative packet $(\mathbf{h}_{k+1}||P_{k+1})$ where the header overhead $\mathbf{h}_{k+1}$ has a minimal length in bits

Algorithm: Small Set of Allowed Coefficients (SSAC)
Input: n, k, m, $GF(q)$, $Q = \{q_0, q_1\}$, (or $Q = \{q_{00}, q_{01}, q_{10}\}$), $\mathbf{Data} = \{(\mathbf{h}_1\|\|P_1), \ldots, (\mathbf{h}_k\|\|P_k)\}$
Output: $(\mathbf{h}_{k+1}\|\|P_{k+1})$
1. Set $\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k)^T$; 2. Set $\mathbf{E} = (\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_k)^T$; where $\mathbf{e}_i$ are CSR forms of $\mathbf{h}_i$ 3. Set $\mathbf{P} = (P_1, P_2, \ldots, P_n)^T$; Repeat 4. Set $\mathbf{w} \leftarrow$ RandomSparseVector(m, n, Q); 5. Find $\mathbf{x}$ such that $\mathbf{w} = \mathbf{x} \cdot \mathbf{E}$; Until found $\mathbf{x}$; 6. Set $\mathbf{e}_{k+1} = \mathbf{w}$; 7. Set $\mathbf{h}_{k+1} =$ CompressedSparseRow$(\mathbf{e}_{k+1})$; 8. Set $P_{k+1} = \mathbf{x} \cdot \mathbf{P}$; 9. Return: $(\mathbf{h}_{k+1}\|\|P_{k+1})$.

We emphasize that our work is the first one that explicitly addresses the efficiency of the header overhead compression algorithm in every intermediate node in conjunction with the number of buffered packets k in that node. In other related works such as [10] and [14] the number of buffered packets in the intermediate nodes as a factor for the efficiency of the algorithm is not addressed at all.

Another important question is the efficiency of the SSAC algorithm for finding a solution of the equation (2). We experimentally measured the number of attempts in the Repeat-Until loop of the SSAC algorithm for the sparsity values $m = 2, 3$ or 4, the number of packets in the range from 16 to 128, and the number of packets in the nodes $k \approx m \log \frac{n}{m}$. The results are given in the next Section.

B. Probability of successful decoding at the destination

Another important aspect in the analysis of SSAC is the probability of successful decoding of the original data. As reported in [7] and [13] the probability of successful decoding depends on the sparsity m, the size of the finite field q and the overhead O (for n encoded packets, the receiver needs $n + O$ packets in order to decode them successfully). Our experiments presented in the next Section confirm that SSAC behaves in the same way as reported in [6], [7], [13]. The probability of successful decoding is lower when using sparse codes compared to dense codes. On the other hand, the probability of successful decoding increases with the overhead and the field size.

IV. Experimental Results

In this Section we illustrate the performance of SSAC and compare it with RLNC and the algorithm based on error correcting codes.

Network coding is usually performed in $GF(16)$ and $GF(256)$, therefore we make experiments for these finite fields for different generation sizes. In order to check the correctness of the presented results, we give the concrete values of the finite fields and the primitive elements that we used. The irreducible polynomial for $GF(16)$ is $i(x) = x^4 + x^3 + 1$. For a two-element Small Set of Allowed Coefficients we choose $Q = \{\mathbf{4}, \mathbf{14}\}$.

The irreducible polynomial for $GF(256)$ is $i(x) = x^8 + x^6 + x^3 + x^2 + 1$. For a two-element Small Set of Allowed Coefficients we choose $Q = \{\mathbf{21}, \mathbf{43}\}$.

First, we investigate the probability of solution existence for different values of m, n and k_{opt} in $GF(16)$ and $GF(256)$. Figure 1 shows the probability of solution existence and the number of attempts in the

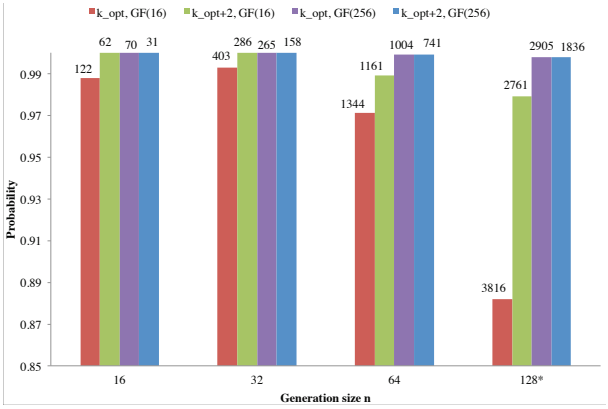


Fig. 1: Probability of solution existence and the number of attempts to find the solution when $m = 2$ for different values of n and k_{opt} in $GF(16)$ and $GF(256)$. Note that the height of the bars denotes the probability, but the values above the bars denote the average number of attempts in SSAC algorithm. Also note that for $n = 128$ in our experiments we took $k = k_{opt} + 8$.

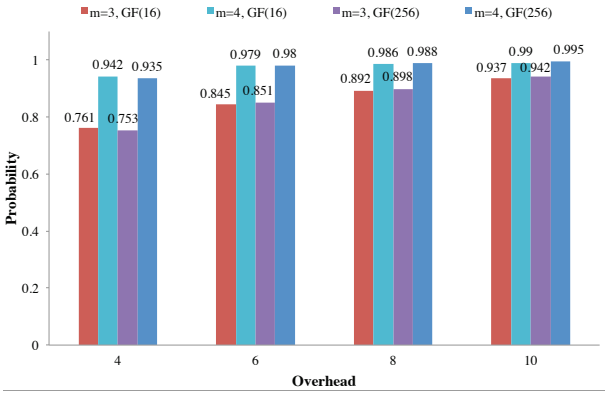


Fig. 2: Probability of a full rank matrix as a function of the overhead when n=16 for different m in $GF(16)$ and $GF(256)$

Repeat-Until loop for the aforementioned parameters for $m = 2$. We see that the probability of solution existence increases with k_{opt} and the size of the field. As we can see that the number of attempts in the Repeat-Until loop varies in the range from 30 up to 3816.

Next we investigate the probability that the received matrix at the destination has a full rank when it has more than n rows, i.e., it has some overhead. Figure 2 shows the probability of a successful decoding as a function of the overhead for $n = 16$ in $GF(16)$ and $GF(256)$ when $m = 3, 4$ and the destination receives overhead of 4, 6, 8 or 10 extra packets. From Figure 2 we can see that probability of having a successful decoding increases with m and the overhead.

We also compare the length of the coding vectors in bits for different schemes. As we stated previously, we compare the performance of SSAC with the most relevant approaches, i.e., traditional RLNC and error correcting codes based approach. Figure 3 to Figure 9 depict the length of the coding vectors in bits versus the generation size n for different m and finite fields. The length of the header overhead increases with the generation size in all three approaches as shown in Figure 3 to Figure 9. However, SSAC achieves around

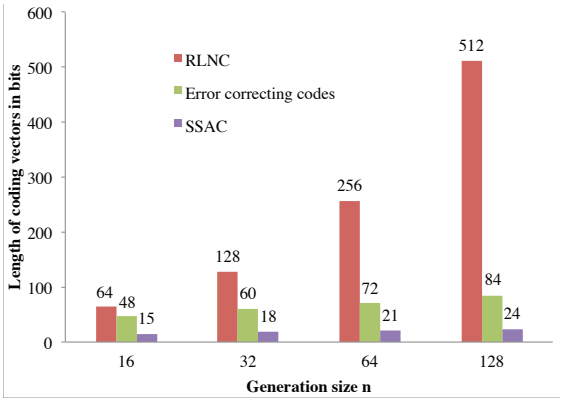


Fig. 3: Length of the compressed coding vectors in bits as a function of the number of packets in a generation n when m=3 in $GF(16)$

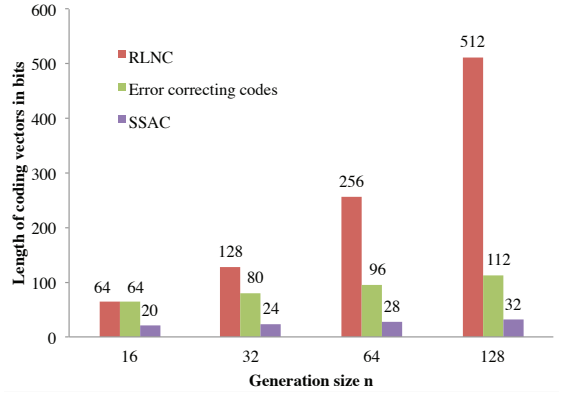


Fig. 4: Length of the compressed coding vectors in bits as a function of the number of packets in a generation n when m=4 in $GF(16)$

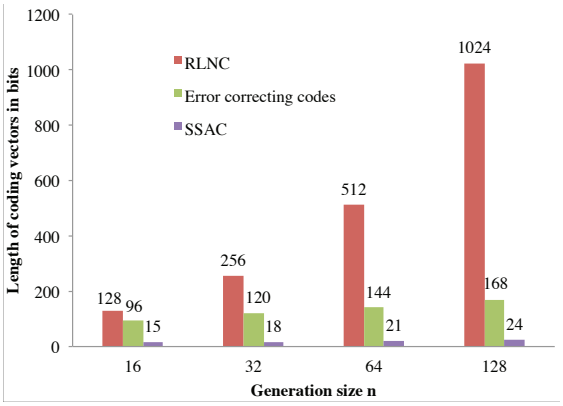


Fig. 5: Length of the compressed coding vectors in bits as a function of the number of packets in a generation n when m=3 in $GF(256)$

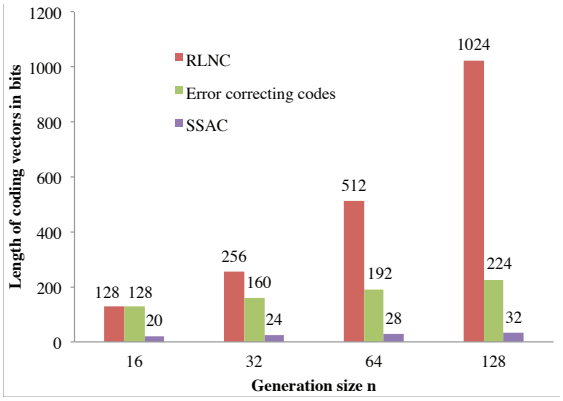


Fig. 6: Length of the compressed coding vectors in bits as a function of the number of packets in a generation n when m=4 in $GF(256)$

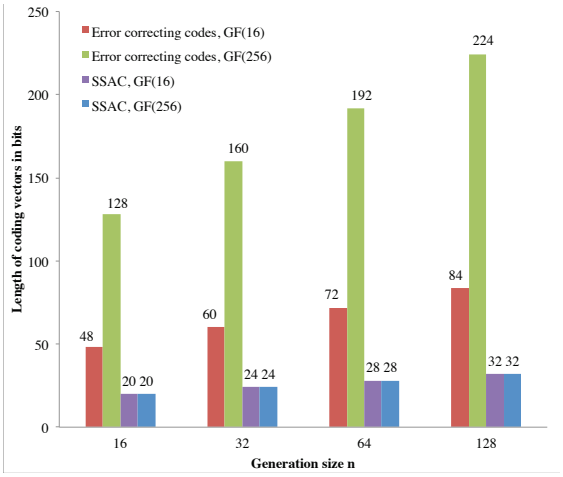


Fig. 7: Length of the compressed coding vectors in bits as a function of the number of packets in a generation n for m=4 in $GF(16)$ and $GF(256)$

42 times shorter header overhead compared to RLNC in $GF(256)$ as presented in Figure 5. Compared to the method with error correcting codes, SSAC produces around 7 times shorter header overheads when m=3 in $GF(256)$. Figure 5 and Figure 6 show that the ratio between the header lengths of SSAC and RLNC is decreasing as m is increasing and the coding is performed in the same finite field. On the other hand, the ratio between the header lengths of SSAC and Error correcting codes approaches is the same for different m in the same finite field.

Figure 7 shows the length of the coding vectors in bits versus the generation size in $GF(16)$ and $GF(256)$ when m is fixed to 4. It is well-known fact that the header overhead increases with the finite field. However, the finite field size does not have an impact on the header overhead when the coding coefficients are generated with SSAC as shown in Figure 7. For instance, the length of the coding vector is the same in $GF(16)$ and $GF(256)$ for the same m. This can be considered as a big advantage of the SSAC algorithm.

Figure 8 and Figure 9 show how the length of the header overhead depends on m. If m increases, then

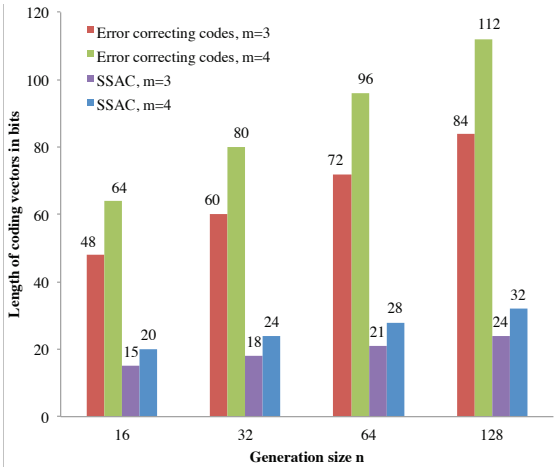


Fig. 8: Length of the compressed coding vectors in bits as a function of the number of packets in a generation n for m=3 and m=4 in $GF(16)$

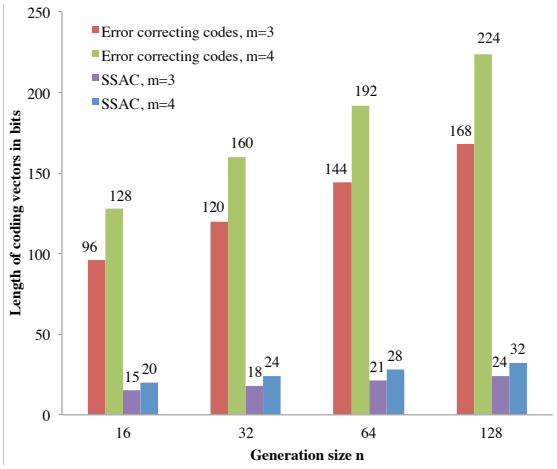


Fig. 9: Length of the compressed coding vectors in bits as a function of the number of packets in a generation n for m=3 and m=4 in $GF(256)$

the length of the header overhead increases as presented in Figure 8 and Figure 9. The step of increasing of the header overhead length in dependence on m is around 8 bits for the SSAC algorithm. On the other hand, this step is significantly bigger for the algorithm based on error correcting codes.

V. Conclusions

In this paper we proposed the algorithm Small Set of Coefficients (SSAC). The SSAC achieves the shortest header overhead in random linear network coding schemes compared to other approaches reported in the literature. We compared the SSAC with RLNC and Error correcting codes. The header overhead length with SSAC is 2 to 7 times shorter than the length achieved by these compression techniques.

We show that the header length does not depend on the size of the finite field where the operations are performed, i.e., it just depends from the number of packets combined m.

As a future work we suggest the following: 1. To establish a precise empirical correlation between the parameters m, n, k, q and the probability of finding solutions of the equation (2); 2. To improve the efficiency of the algorithm SSAC by replacing the randomized search approach in Step 4 with a faster direct algorithm for finding suitable $\mathbf{w}$ and $\mathbf{x}$ in equation (2); 3. To introduce and examine a realistic metric of energy consumption in the source and intermediate nodes for producing and transmitting the headers; and 4. To investigate the relation between the elements in the set $|Q|$ and the efficiency of the algorithm.

REFERENCES

[1] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.

[2] Khanh Do Ba, Piotr Indyk, Eric Price, and David P. Woodruff. Lower bounds for sparse recovery. *CoRR*, abs/1106.0365, 2011.

[3] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, 2 edition, 1994.

[4] Johannes Blmer, Richard Karp, and Emo Welzl. The rank of sparse random matrices over finite fields, 1997.

[5] Cheng-Chih Chao, Ching-Chun Chou, and Hung-Yu Wei. Pseudo random network coding design for IEEE 802.16m enhanced multicast and broadcast service. In *VTC Spring*, pages 1–5. IEEE, 2010.

[6] S. Feizi, D.E. Lucani, C.W. Sorensen, A. Makhdoumi, and M. Medard. Tunable sparse network coding for multicast networks. In *Network Coding (NetCod), 2014 International Symposium on*, pages 1–6, June 2014.

[7] Janus Heide, Morten Videbæk Pedersen, Frank H. P. Fitzek, and Muriel Médard. On code parameters and coding vector representation for practical RLNC. In *ICC*, pages 1–5. IEEE, 2011.

[8] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong. A random linear network coding approach to multicast. *IEEE Transactions on Information Theory*, 52(10):4413–4430, 2006.

[9] Ralf Koetter and Frank R. Kschischang. Coding for errors and erasures in random network coding. *IEEE Transactions on Information Theory*, 54(8):3579–3591, 2008.

[10] Shizheng Li and Aditya Ramamoorthy. Improved compression of network coding vectors using erasure decoding and list decoding. *IEEE Communications Letters*, 14(8):749–751, 2010.

[11] Zimu Liu, Chuan Wu, Baochun Li, and Shuqiao Zhao. UUSee: Large-scale operational on-demand streaming with random network coding. In *INFOCOM*, pages 2070–2078. IEEE, 2010.

[12] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TAG: A tiny AGgregation service for ad-hoc sensor networks. In *OSDI 2002*, pages 1–16, 2002.

[13] Payam Pakzad, Christina Fragouli, and Amin Shokrollahi. Coding schemes for line networks. In *Proc. ISIT*, 2005.

[14] Mahdi Jafari Siavoshani, Lorenzo Keller, Christina Fragouli, and Katerina J. Argyraki. Compressed network coding vectors. In *ISIT*, pages 109–113. IEEE, 2009.

[15] Nikolaos Thomos and Pascal Frossard. Toward one symbol network coding vectors. *IEEE Communications Letters*, 16(11):1860–1863, 2012.

General Sub-packetized Access-Optimal Regenerating Codes
Katina Kralevska, Danilo Gligoroski, and Harald Øverby
IEEE Communications Letters, vol. 20, issue 7, pp. 1281-1284, 2016

Paper 4

General Sub-packetized Access-Optimal Regenerating Codes

Katina Kralevska, Danilo Gligoroski, and Harald Øverby

Abstract

This paper presents a novel construction of $(n, k, d = n-1)$ access-optimal regenerating codes for an arbitrary sub-packetization level α for exact repair of any systematic node. We refer to these codes as general sub-packetized because we provide an algorithm for constructing codes for any α less than or equal to $r^{\lceil \frac{k}{r} \rceil}$ where $\frac{k}{r}$ is not necessarily an integer. This leads to a flexible construction of codes for different code rates compared to existing approaches. We derive the lower and the upper bound of the repair bandwidth. The repair bandwidth depends on the code parameters and α. The repair process of a failed systematic node is linear and highly parallelized, which means that a set of $\lceil \frac{\alpha}{r} \rceil$ symbols is independently repaired first and used along with the accessed data from other nodes to recover the remaining symbols.

***Index Terms*- Minimum storage regenerating codes, sub-packetization, access-optimal**

I. Introduction

Erasure coding is becoming an attractive technique for data protection since it offers the same level of reliability with significantly less storage overhead compared to replication [1]. Apart from the reliability and the storage overhead, there are other desirable features in a distributed storage system such as low repair bandwidth and access-optimality. Repair bandwidth is the amount of transferred data during a repair process. Access-optimality is achieved when the amount of accessed and transferred data during the repair process is equal.

Dimakis et al. introduced regenerating codes that significantly reduce the repair bandwidth [2]. Under an (n, k, d) regenerating code, a file of M symbols from a finite field $\mathbf{F}_q$ is divided into k fragments, each of size $\alpha = \frac{M}{k}$ symbols, which are further encoded into n fragments using an (n, k) MDS (Maximum-Distance Separable) code. The parameter α, termed as a sub-packetization level of the code, represents the minimum dimension over which operations are performed. The data from a failed node is recovered by transferring β symbols from each d non-failed nodes. Thus, the repair bandwidth γ is equal to $d\beta$ where $\alpha \leq d\beta \ll M$. Dimakis et al. [2] showed the existence of minimum storage regenerating (MSR) codes that attain the minimum storage point of the optimal tradeoff curve between the storage and the repair bandwidth, i.e.,

$$(\alpha_{MSR}, \gamma_{MSR}^{min}) = (\frac{M}{k}, \frac{M}{k}\frac{n-1}{n-k}). \tag{1}$$

The repair bandwidth is minimized when all $d = n-1$ non-failed nodes transmit a fraction of $1/r$ of the stored data.

Several exact repair MSR codes, which are characterized by the repaired data being exactly the same as the lost data, have been suggested. Tamo et al. proposed optimal MSR codes for α equal to r^k known as zigzag codes [3]. Furthermore, they showed that α of access-optimal MSR codes for repair of any systematic node is $r^{\frac{k}{r}}$ [4]. The codes presented in [5] and [6] meet this condition. An essential condition for the code construction in [6] is that $m = \frac{k}{r}$ has to be an integer $m \geq 1$ where k is set to rm and α to r^m. Wang et al. constructed codes that optimally repair any systematic or parity node for α equal to

Manuscript received March 18, 2016; revised April 25, 2016; accepted April 25, 2016. Date of publication May 2, 2016; date of current version July 8, 2016. The associate editor coordinating the review of this letter and approving it for publication was R. D. Souza. The authors are with the Department of Telematics, NTNU, Norwegian University of Science and Technology, Trondheim 7491, Norway (e-mail: katinak@item.ntnu.no; danilog@item.ntnu.no; haraldov@item.ntnu.no).

r^{k+1} [7]. High-rate MSR codes with a polynomial sub-packetization level are proposed in [8]. However, our work focuses only on code constructions for optimal repair of any systematic node.

MSR codes are optimal in terms of storage, reliability and repair bandwidth, but not I/O. Implementing MSR codes with a sub-packetization level of $r^{\frac{k}{r}}$ may not be practical for storage systems that serve applications with a large number of user requests or perform intensive computations. Thus, having an algorithm for constructing MSR codes for any combination of n, k and α that are simultaneously optimal in terms of storage, reliability, repair bandwidth and I/O is an important problem that is solved in this work.

Our Contribution: This paper presents a novel construction of $(n, k, d = n - 1)$ access-optimal regenerating codes for an arbitrary sub-packetization level for exact repair of any systematic node. The codes have the following properties: 1. MDS; 2. Systematic; 3. Flexible sub-packetization level; 4. Minimum repair bandwidth for every α including the lower bound (1) when α is $r^{\lceil \frac{k}{r} \rceil}$; 5. Access-optimality; 6. Fast decoding. To the best of the authors' knowledge, these are the first code constructions for an arbitrary α. Motivated by the code construction in [6], we construct general codes where $\frac{k}{r}$ does not need to be an integer and α is not exclusively equal to $r^{\frac{k}{r}}$. For instance, the code $(14, 10)$ that is deployed in the data-warehouse cluster of Facebook [9] is out of the scope of applicability with the current proposals in [4]–[6], because $\frac{k}{r} = 2.5$ is a non-integer. However, the presented algorithm constructs an $(14, 10, 13)$ code that reduces the repair bandwidth for any systematic node by 67.5% when α is $r^{\lceil \frac{k}{r} \rceil} = 64$ compared to an $(14, 10)$ RS code. The repair process is linear and highly parallelized.

II. A General $(n, k, d = n - 1)$ Code Construction

Consider a file of size $M = k\alpha$ symbols from a finite field $\mathbf{F}_q$ stored in k systematic nodes d_j of capacity α symbols.

We define a systematic MDS code in the following way: The basic data structure component is an index array of size $\alpha \times k$ where $\alpha \leq r^{\lceil \frac{k}{r} \rceil}$ and $n = k + r$, $P = ((i, j))_{\alpha \times k}$. We use r such index arrays $P_1, \ldots, P_r$. The elements $p_{i,1}$, $i = 1, \ldots, \alpha$, in p_1 are a linear combination only of the symbols with indexes present in the rows of P_1. In the initialization phase, additional $\lceil \frac{k}{r} \rceil$ columns with pairs $(0, 0)$ are added to $P_2, \ldots, P_r$. The goal of the algorithm is to replace those zero pairs with concrete (i, j) pairs so that the code is access-optimal for a given sub-packetization level α. The value of α determines two phases of the algorithm. In the first phase, the indexes (i, j) that replace the $(0, 0)$ pairs are chosen such that both Condition 1 and Condition 2 are satisfied (defined further in this section). The first phase starts with a granulation level parameter called run that is initialized with the value $\lceil \frac{\alpha}{r} \rceil$. This parameter affects how the indexes (i, j) are chosen and with every round the granulation level decreases by a factor r. Once the granulation level becomes equal to 1 and there are still $(0, 0)$ pairs that have to get some value (i, j), the second phase starts where the remaining indexes are chosen such that only Condition 2 is satisfied.

A high level description of the proposed algorithm is given in Alg. 1, while a detailed one in Alg. 2.

Algorithm 1 High level description of an algorithm for generating general sub-packetized, access-optimal regenerating codes

1: Initialize the index arrays $P_1, \ldots, P_r$;
2: # Phase 1
3: Set the granulation level $run \leftarrow \lceil \frac{\alpha}{r} \rceil$
4: **repeat**
5: Replace $(0, 0)$ pairs with indexes (i, j) such that both Condition 1 and Condition 2 are satisfied;
6: Decrease the granulation level run by a factor r.
7: **until** the granulation level $run > 1$
8: # Phase 2
9: If there are still $(0, 0)$ indexes that have to get some value (i, j), choose them such that only Condition 2 is satisfied;
10: Return the index arrays $P_1, \ldots, P_r$;

Once the index arrays $P_1, \ldots, P_r$ are determined, the symbols $p_{i,l}$ in the parity nodes, $1 \leq i \leq \alpha$ and $1 \leq l \leq r$, are generated as a combination of the elements a_{j_1,j_2} where the pair (j_1, j_2) is in the i-th row of the index array P_l, i.e.,

$$p_{i,l} = \sum c_{l,i,j} a_{j_1,j_2}. \tag{2}$$

The linear relations have to guarantee an MDS code, i.e., to guarantee that the entire information can be recovered from any k nodes (systematic or parity). We use the following terms and variables:

Algorithm 2 Algorithm to generate the index arrays
Input: n, k, α;
Output: Index arrays $P_1, \ldots, P_r$.

Initialization: $P_1, \ldots, P_r$ are initialized as index arrays $P = ((i,j))_{\alpha \times k}$;
Append $\lceil \frac{k}{r} \rceil$ columns to $P_2, \ldots, P_r$ all initialized to $(0,0)$;
Set $portion \leftarrow \lceil \frac{\alpha}{r} \rceil$;
Set $ValidPartitions \leftarrow \emptyset$;
Set $j \leftarrow 0$;
# Phase 1
repeat
    Set $j \leftarrow j + 1$;
    Set $\nu \leftarrow \lceil \frac{j}{r} \rceil$;
    Set $run \leftarrow \lceil \frac{\alpha}{r^\nu} \rceil$;
    Set $step \leftarrow \lceil \frac{\alpha}{r} \rceil - run$;
    $\mathcal{D}_{d_j} = ValidPartitioning(ValidPartitions,\ k,\ r,\ portion,\ run,\ step,\ J_\nu)$;
    Set $ValidPartitions = ValidPartitions \cup \mathcal{D}_{d_j}$;
    Determine one $D_{\rho,d_j} \in \mathcal{D}_{d_j}$ such that its elements correspond to row indexes in the $(k+\nu)$-th column in one of the arrays $P_2, \ldots, P_r$, that are all zero pairs $(0,0)$;
    The indexes in D_{ρ,d_j} are the row positions where the pairs (i,j) with indexes $i \in \mathcal{D} \setminus D_{\rho,d_j}$ are assigned in the $(k+\nu)$-th column of $P_2, \ldots, P_r$;
until $(run > 1)$ AND $(j \neq 0 \mod r)$
# Phase 2
while $j < k$ **do**
    Set $j \leftarrow j + 1$;
    Set $\nu \leftarrow \lceil \frac{j}{r} \rceil$;
    Set $run \leftarrow 0$;
    $\mathcal{D}_{d_j} = ValidPartitioning(ValidPartitions,\ k,\ r,\ portion,\ run,\ step,\ J_\nu)$;
    Set $ValidPartitions = ValidPartitions \cup \mathcal{D}_{d_j}$;
    Determine one $D_{\rho,d_j} \in \mathcal{D}_{d_j}$ such that its elements correspond to row indexes in the $(k+\nu)$-th column in one of the arrays $P_2, \ldots, P_r$, that are all zero pairs $(0,0)$;
    The indexes in D_{ρ,d_j} are the row positions where the pairs (i,j) with indexes $i \in \mathcal{D} \setminus D_{\rho,d_j}$ are assigned in the $(k+\nu)$-th column of $P_2, \ldots, P_r$;
end while
Return $P_1, \ldots, P_r$.

- The set $Nodes = \{d_1, \ldots, d_k\}$ of k systematic nodes is partitioned in $\lceil \frac{k}{r} \rceil$ disjunctive subsets $J_1, J_2, \ldots, J_{\lceil \frac{k}{r} \rceil}$ where $|J_\nu| = r$ (if r does not divide k then $J_{\lceil \frac{k}{r} \rceil}$ has $k \mod r$ elements) and $Nodes = \cup_{\nu=1}^{\lceil \frac{k}{r} \rceil} J_\nu$. In general, this partitioning can be any random permutation of k nodes. Without

loss of generality we use the natural ordering as follows: $J_1 = \{d_1, \ldots, d_r\}$, $J_2 = \{d_{r+1}, \ldots, d_{2r}\}$, $\ldots$, $J_{\lceil \frac{k}{r} \rceil} = \{d_{\lfloor \frac{k}{r} \rfloor \times r+1}, \ldots, d_k\}$.

- Each node d_j consists of an indexed set of α symbols $\{a_{1,j}, a_{2,j}, \ldots, a_{\alpha,j}\}$.
- $portion = \lceil \frac{\alpha}{r} \rceil$: The set of all symbols in d_j is partitioned in disjunctive subsets where at least one subset has $portion$ number of elements.
- $run = \lceil \frac{\alpha}{r^\nu} \rceil$, for values of $\nu \in \{1, \ldots, \lceil \frac{k}{r} \rceil\}$.
- $step = \lceil \frac{\alpha}{r} \rceil - run$: For the subsequent $(k+\nu)$-th column, where $\nu \in \{1, \ldots, \lceil \frac{k}{r} \rceil\}$, the scheduling of the indexes corresponding to the nodes in J_ν is done in subsets of indexes from a valid partitioning.
- *A valid partitioning* $\mathcal{D}_{d_j} = \{D_{1,d_j}, \ldots, D_{r,d_j}\}$ of a set of indexes $D = \{1, \ldots, \alpha\}$, where the $i-$th symbol in d_j is indexed by i in D, is a partitioning in r disjunctive subsets $D_{d_j} = \cup_{\rho=1}^{r} D_{\rho,d_j}$. If r divides α, then the valid partitioning for all nodes in J_ν is equal. If r does not divide α, then the valid partitioning has to contain at least one subset D_{ρ,d_j} with $portion$ elements that correspond to the row indexes in the $(k+\nu)$-th column in one of the arrays $P_2, \ldots, P_r$ that are all zero pairs.
- *Condition 1*: At least one subset D_{ρ,d_j} has $portion$ elements with runs of run consecutive elements separated with a distance between the indexes equal to $step$. The elements of that subset correspond to the row indexes in the $(k+\nu)$-th column in one of the arrays $P_2, \ldots, P_r$ that are all zero pairs. The distance between two elements in one node is computed in a cyclical manner such that the distance between the elements $a_{\alpha-1}$ and a_2 is 2.
- *Condition 2*: A necessary condition for the valid partitioning to achieve the lowest possible repair bandwidth is $D_{d_{j_1}} = D_{d_{j_2}}$ for all d_{j_1} and d_{j_2} in J_ν and $D_{\rho,d_{j_1}} \neq D_{\rho,d_{j_2}}$ for all d_{j_1} and d_{j_2} systematic nodes in the system. If $portion$ divides α, then D_{ρ,d_j} for all d_j in J_ν are disjunctive, i.e., $D = \cup_{j=1}^{r} D_{\rho,d_j} = \{1, \ldots, \alpha\}$.

Algorithm 3 $ValidPartitioning$
Input:$ValidPartitions, k, r, portion, run, step, J_\nu$;
Output: $\mathcal{D}_{d_j} = \{D_{1,d_j}, \ldots, D_{r,d_j}\}$.

1: Set $D = \{1, \ldots, \alpha\}$;
2: **if** $run \neq 0$ **then**
3: Find $\mathcal{D}_{d_j}$ that satisfies Condition 1 and Condition 2;
4: **else**
5: Find $\mathcal{D}_{d_j}$ that satisfies Condition 2;
6: **end if**
7: Return $\mathcal{D}_{d_j}$;

Alg. 4 shows the repair of a systematic node where the systematic and the parity nodes are global variables. A set of $\lceil \frac{\alpha}{r} \rceil$ symbols is accessed and transferred from each $n-1$ non-failed nodes. If $\alpha \neq r^{\lceil \frac{k}{r} \rceil}$, then additional elements may be required as described in Step 4. Note that a specific element is transferred just once and stored in a buffer. For every subsequent use of that element, the element is read from the buffer and further transfer operation is not required. The repair process is highly parallelized, because a set of $\lceil \frac{\alpha}{r} \rceil$ symbols is independently and in parallel repaired in Step 2 and then the remaining symbols are recovered in parallel in Step 5.

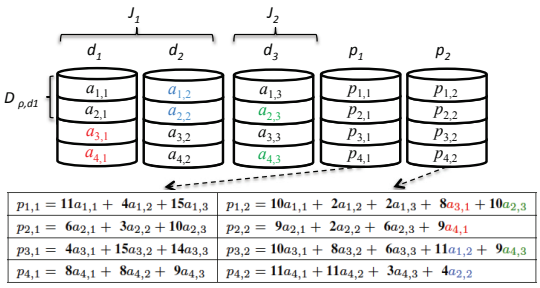
Proposition 1: The repair bandwidth for a single systematic node γ is bounded between the following lower and upper bounds:

$$\frac{(n-1)}{r} \leq \gamma \leq \frac{(n-1)}{\alpha} \lceil \frac{\alpha}{r} \rceil + \frac{(r-1)}{\alpha} \lceil \frac{\alpha}{r} \rceil \lceil \frac{k}{r} \rceil. \tag{3}$$

Proof: Note that we read in total $k\lceil \frac{\alpha}{r} \rceil$ elements in Step 1 of Alg. 4. Additionally, $(r-1)\lceil \frac{\alpha}{r} \rceil$ elements are read in Step 3. Assuming that we do not read more elements in Step 4, we determine the lower bound as $k\lceil \frac{\alpha}{r} \rceil + (r-1)\lceil \frac{\alpha}{r} \rceil = (n-1)\lceil \frac{\alpha}{r} \rceil$ elements, i.e., the lower bound is $\frac{(n-1)}{\alpha}\lceil \frac{\alpha}{r} \rceil$ (since every element has

Algorithm 4 Repair of a systematic node d_l
Input: l;
Output: d_l.

1: Access and transfer $(k-1)\lceil\frac{\alpha}{r}\rceil$ elements $a_{i,j}$ from all $k-1$ non-failed systematic nodes and $\lceil\frac{\alpha}{r}\rceil$ elements $p_{i,1}$ from p_1 where $i \in D_{\rho,d_l}$;
2: Repair $a_{i,l}$ where $i \in D_{\rho,d_l}$
3: Access and transfer $(r-1)\lceil\frac{\alpha}{r}\rceil$ elements $p_{i,j}$ from $p_2,\ldots,p_r$ where $i \in D_{\rho,d_l}$;
4: Access and transfer from the systematic nodes the elements $a_{i,j}$ listed in the $i-$th row of the arrays $P_2,\ldots,P_r$ where $i \in D_{\rho,d_l}$ that have not been read in Step 1;
5: Repair $a_{i,l}$ where $i \in \mathcal{D} \setminus D_{\rho,d_l}$;



$p_{1,1} = 11a_{1,1} + 4a_{1,2} + 15a_{1,3}$	$p_{1,2} = 10a_{1,1} + 2a_{1,2} + 2a_{1,3} + 8a_{3,1} + 10a_{2,3}$
$p_{2,1} = 6a_{2,1} + 3a_{2,2} + 10a_{2,3}$	$p_{2,2} = 9a_{2,1} + 2a_{2,2} + 6a_{2,3} + 9a_{4,1}$
$p_{3,1} = 4a_{3,1} + 15a_{3,2} + 14a_{3,3}$	$p_{3,2} = 10a_{3,1} + 8a_{3,2} + 6a_{3,3} + 11a_{1,2} + 9a_{4,3}$
$p_{4,1} = 8a_{4,1} + 8a_{4,2} + 9a_{4,3}$	$p_{4,2} = 11a_{4,1} + 11a_{4,2} + 3a_{4,3} + 4a_{2,2}$

Fig. 1: An MDS array code with 3 systematic and 2 parity nodes for $\alpha = 4$. The elements presented in colors are scheduled as additional elements in p_2.The coefficients are from $\mathbf{F}_{16}$ with irreducible polynomial $x^4 + x^3 + 1$.

a size of $\frac{1}{\alpha}$). To derive the upper bound, we assume that we read all elements $a_{i,j}$ from the extra $\lceil\frac{k}{r}\rceil$ columns of the arrays $P_2,\ldots,P_r$ in Step 4. Thus, the upper bound is $\frac{(n-1)}{\alpha}\lceil\frac{\alpha}{r}\rceil + \frac{(r-1)}{\alpha}\lceil\frac{\alpha}{r}\rceil\lceil\frac{k}{r}\rceil$. ■

The optimality of the proposed code construction is captured in the following Proposition.

Proposition 2: If r divides α, then the indexes (i,j) of the elements $a_{i,j}$ where $i \in \mathcal{D} \setminus D_{\rho,d_j}$ for each group of r systematic nodes are scheduled in one of the $\lceil\frac{k}{r}\rceil$ additional columns in the index arrays $P_2,\ldots,P_r$.

Next we show that there always exists a set of non-zero coefficients from $\mathbf{F}_q$ in the linear combinations so that the code is MDS. We adapt Theorem 4.1 from [6] as follows:

Theorem 1: There exists a choice of non-zero coefficients $c_{l,i,j}$ where $l = 1,\ldots,r$, $i = 1,\ldots,\alpha$ and $j = 1,\ldots,k$ from $\mathbf{F}_q$ such that the code is MDS if $q \geq \binom{n}{k}r\alpha$.

Proof: The system of linear equations in (2) defines a system of $r \times \alpha$ linear equations with $k \times \alpha$ variables. A repair of one failed node is given in Alg. 4, but for the sake of this proof, we explain the repair by discussing the solutions of the system of equations. When one node has failed, we have an overdefined system of $r \times \alpha$ linear equations with α unknowns. In general this can lead to a situation where there is no solution. However, since the values in system (2) are obtained from the values of the lost node, we know that there exists one solution. Thus, solving this system of $r \times \alpha$ linear equations with an overwhelming probability gives a unique solution, i.e., the lost node is recovered. When 2 nodes have failed, we have a system of $r \times \alpha$ linear equations with 2α unknowns. The same discussion for the overdefined system applies here. The most important case is when $r = n - k$ nodes have failed. In this case, we have a system of $r \times \alpha$ linear equations with $r \times \alpha$ unknowns. If the size of the finite field $\mathbf{F}_q$ is large enough, i.e., $q \geq \binom{n}{k}r\alpha$, as it is shown in Theorem 4.1 in [6], the system has a unique solution, i.e., the file M can be collected from any k nodes. ■

III. Code Examples

Let us take the $(5, 3, 4)$ code. We show a code construction for the optimal sub-packetization level $\alpha = 2^{\lceil \frac{3}{2} \rceil} = 4$.

The following requirements have to be satisfied for the code to be an access-optimal MDS code that achieves the lower bound of the repair bandwidth for any systematic node:

- $M = k\alpha = 12$ symbols,
- Repair a failed systematic node by accessing and transferring $\lceil \frac{\alpha}{r} \rceil = 2$ symbols from the remaining $d = 4$ nodes,
- Reconstruct the data from any 3 nodes.

The systematic nodes d_1, d_2, d_3 and the parity nodes p_1, p_2 are shown in Fig. 1. The file size is 12 symbols, where each node stores $\alpha = 4$ symbols. The elements of p_1 are linear combinations of the row elements from the systematic nodes multiplied by coefficients from $\mathbf{F}_{16}$. The elements of p_2 are obtained by adding extra symbols to the row sum. We next show the scheduling of an element $a_{i,j}$ from a specific d_j where $i \in \mathcal{D} \setminus D_{\rho,d_j}$ at $portion = 2$ positions in the i-th row, $i \in D_{\rho,d_j}$, and the $(3 + \nu)$-th column, $\nu = 1, 2$, of P_2. We follow the steps in Alg. 2 and give a brief description:
1. Initialize P_1 and P_2 as arrays $P = ((i, j))_{4\times 3}$.
2. Append additional 2 columns to P_2 initialized to $(0, 0)$.
3. Set $portion = 2$ and $ValidPartitions = \emptyset$.
4. For the nodes d_1, d_2 that belong to J_1, run is equal to 2 and $step$ to 0. While run is equal to 1 and $step$ to 1 for the node d_3 that belongs to J_2.
5. Alg. 3 gives $D_{d_1} = \{\{1, 2\}, \{3, 4\}\}$. Following step 14 in Alg. 2, the first 2 zero pairs in the 5-th column of P_2 with 0 distance between them are at the positions $(i, 5)$ where $i = 1, 2$. Thus, $D_{\rho,d_1} = \{1, 2\}$. We follow the same logic to obtain $D_{\rho,d_2} = \{3, 4\}$ and $D_{\rho,d_3} = \{1, 3\}$. Next we schedule the elements of d_j with i indexes that are not elements of D_{ρ,d_j} (represented in colors in Fig. 1) in the $i-$th row of P_2 where $i \in D_{\rho,d_j}$. The $i-$th index of $a_{3,1}$ and $a_{4,1}$ does not belong to D_{ρ,d_1} so these elements are scheduled at the positions $(1, 5)$ and $(2, 5)$ of P_2. We add the elements $a_{1,2}, a_{2,2}$ from d_2 in the 3-rd and the 4-th row respectively, while we add the elements $a_{2,3}$ and $a_{4,3}$ from d_3 in the 1-st and 3-rd row respectively. The symbols in the parity nodes are obtained as linear combinations of the row elements in the parity arrays. For instance, $p_{1,2}$ is a linear combination of the elements in the first row from all systematic nodes, $a_{3,1}$ and $a_{2,3}$.

We next show how to repair the node d_1 following Alg. 4. First, we repair the elements $a_{1,1}, a_{2,1}$. Thus, we access and transfer $a_{1,2}$, $a_{2,2}$, $a_{1,3}$ and $a_{2,3}$ from d_2 and d_3 and $p_{1,1}$, $p_{2,1}$ from p_1. In order to recover $a_{3,1}, a_{4,1}$, we need to access and transfer $p_{1,2}$ and $p_{2,2}$ from p_2. Hence, the data from d_1 is recovered by accessing and transferring in total 8 elements from 4 non-failed nodes. Exactly the same amount of data, 8 symbols, is needed to repair d_2 or d_3. Thus, the average repair bandwidth, defined as the ratio of the total repair bandwidth to repair all systematic nodes to the file size M, is equal to 2 symbols.

A. Performance Analysis

Another code discussed in this section is $(14, 10, 13)$ with different α. Fig. 2 shows the relation between the average repair bandwidth and α. For an RS code, α is 1 and the average repair bandwidth is equal to the file size. A Hitchhiker code [9] for $\alpha = 2$ reduces the repair bandwidth by 35% compared to the RS code. The remaining values of the average repair bandwidth are for the codes constructed with the algorithms presented in Section II. We observe that the lower bound of the repair bandwidth that is 3.25 is achieved for $\alpha = r^{\lceil \frac{k}{r} \rceil} = 64$. As we can see the repair bandwidth decreases as α increases.

IV. Conclusions

We presented a general construction of access-optimal regenerating codes that reach the lower bound of the repair bandwidth for $\alpha = r^{\lceil \frac{k}{r} \rceil}$, while the repair bandwidth is as close as possible to the lower bound when $\alpha < r^{\lceil \frac{k}{r} \rceil}$. The repair process of a systematic node is linear and highly parallelized.

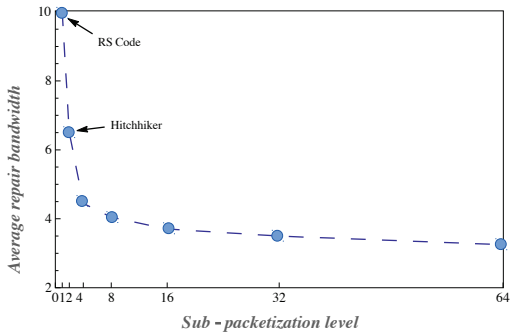


Fig. 2: Average repair bandwidth for any systematic node for different sub-packetization levels α for an (14, 10, 13) code

REFERENCES

[1] H. Weatherspoon and J. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *Proc. 1st Int. Workshop on Peer-to-Peer Systems*, 2002, pp. 328–338.

[2] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sept. 2010.

[3] I. Tamo, Z. Wang, and J. Bruck, "Zigzag codes: Mds array codes with optimal rebuilding," *IEEE Trans. Inf. Theory*, vol. 59, no. 3, pp. 1597–1616, March 2013.

[4] I. Tamo, Z. Wang, and J. Bruc, "Access vs. bandwidth in codes for storage," in *Proc. IEEE Int. Symp. Inf. Theory*, 2012, pp. 1187–1191.

[5] V. R. Cadambe, C. Huang, J. Li, and S. Mehrotra, "Polynomial length mds codes with optimal repair in distributed storage," in *Proc. 45th Asilomar Conf. Signals, Syst., Comp.*, 2011, pp. 1850–1854.

[6] G. K. Agarwal, B. Sasidharan, and P. V. Kumar, "An alternate construction of an access-optimal regenerating code with optimal sub-packetization level," in *Proc. 21st Nat. Conf. Comm.*, 2015, pp. 1–6.

[7] Z. Wang, I. Tamo, and J. Bruck, "On codes for optimal rebuilding access," in *Proc. 49th Annual Allerton Conf. Comm., Control, Comp.,*, 2011, pp. 1374–1381.

[8] B. Sasidharan, G. K. Agarwal, and P. V. Kumar, "A high-rate msr code with polynomial sub-packetization level," in *Proc. IEEE Int. Symp. Inf. Theory*, 2015, pp. 2051–2055.

[9] K. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A "hitchhiker's" guide to fast and efficient data reconstruction in erasure-coded data centers," in *Proc. ACM SIGCOMM Conf.*, 2014, pp. 331–342.

HashTag Erasure Codes: From Theory to Practice
Katina Kralevska, Danilo Gligoroski, Rune E.Jensen, and Harald Øverby
Submitted to IEEE Transactions on Big Data

Paper 5

HashTag Erasure Codes: From Theory to Practice

Katina Kralevska, Danilo Gligoroski, Rune E. Jensen, and Harald Øverby

Abstract

Erasure coding has been increasingly deployed as an alternative to data-replication for fault-tolerance in distributed-storage systems. Conventional erasure codes such as Reed-Solomon provide savings in the storage space, but at the cost of higher repair bandwidth and more complex computations than replication. Minimum-Storage Regenerating (MSR) codes have emerged as a viable alternative to Reed-Solomon codes as they minimize the repair bandwidth while still being optimal in terms of reliability and storage overhead. Although several MSR code constructions exist, so far they have not been practically implemented. One of the main reasons for their practical abandonment is that existing MSR code constructions imply much bigger number of I/O operations than RS codes.

In this paper, we analyze high-rate MDS codes that are simultaneously optimal in terms of storage, reliability, I/O operations and repair-bandwidth for single and multiple failures of the systematic nodes. The codes were recently introduced in [1] without any specific name. Due to the resemblance between the hashtag sign # and the procedure of the construction of these codes, we call them here *HashTag Erasure Codes (HTECs)*.

HTECs provide the lowest data-read and data-transfer for an arbitrary sub-packetization level α where $\alpha \leq r^{\left\lceil \frac{k}{r} \right\rceil}$ among all existing solutions for distributed storage. The repair process is linear and highly parallel. Additionally, we show that HTECs are the first high-rate MDS codes that reduce the repair bandwidth for more than one failure.

Index Terms

Distributed storage, MDS erasure codes, sub-packetization, access-optimal, I/O, single and multiple failures.

I. Introduction

ERASURE coding has become a viable alternative to replication as it provides the same level of reliability with significantly less storage overhead [2]. When replication is used, the data is available as long as at least one copy still exists. The storage overhead of storing one extra replica is 100%, while it is 200% for 2 replicas and so forth. Therefore, replication is not suitable for large-scale storage systems. Its high storage overhead implies a high hardware cost (disk drives and associated equipment), as well as high operational costs that include building space, power, cooling, maintenance, etc.

Compared to replication, traditional erasure codes reduce the storage overhead, but at a higher repair cost that is expressed through a high repair bandwidth (the amount of data transferred during the repair process), excessive input and output operations (I/Os) and expensive computations. Practical implementations of erasure codes in distributed storage systems such as Google File System (GFS) [3] and Hadoop Distributed File System (HDFS) [4] require the erasure codes to be Maximum Distance Separable (MDS), high-rate and repair-efficient. Two primary metrics that determine the repair efficiency of a code are the amount of accessed data (*data-read*) from the non-failed nodes and the amount of transferred data (*repair bandwidth*). We are interested in codes where these two metrics are *minimized* and *equal* at the same time. The amount of data transferred characterizes the network usage, while the accessed data corresponds to disk I/O operations. The number of I/Os is an important parameter in storage systems especially for applications that serve a large number of user requests or perform data intensive computations where I/Os are becoming the primary bottleneck. There are two main types of I/Os: sequential and random operations. *Sequential*

K. Kralevska, D. Gligoroski and H. Øverby are with the Department of Telematics, NTNU, Norwegian University of Science and Technology, Trondheim 7491, Norway. E-mail: {katinak, danilog, haraldov}@item.ntnu.no.

R. E. Jensen is with the Department of Computer and Information Science, NTNU, Norwegian University of Science and Technology, Trondheim 7491, Norway. E-mail: runeerle@idi.ntnu.no.

operations access locations on the storage device in a contiguous manner, while *random operations* access locations on the storage device in a non-contiguous manner.

Reed-Solomon (RS) codes [5] are a well-known representative of traditional *MDS* codes. Under an (n, k) RS code, a file of M symbols is stored across n nodes with equal capacity of $\frac{M}{k}$ symbols. The missing/unavailable data from one node can be recovered from any k out of n nodes. Thus, a transfer of $k \times \frac{M}{k} = M$ (the whole file of M symbols) is needed in order to repair $\frac{1}{k}$ of the file. RS codes tolerate up to $n-k$ failures without any permanent data-loss. In general, the repair bandwidth and the number of I/Os are k times higher with RS codes than with replication. During the reconstruction process with RS codes, the entire data from k nodes has to be read and hence trivially, the reads are sequential.

A powerful class of erasure codes that optimizes for storage and repair bandwidth costs has been proposed in [6]. Minimum Bandwidth Regenerating (MBR) codes are optimal in terms of the repair bandwidth, while *Minimum Storage Regenerating (MSR)* codes in terms of the storage. MSR codes possess all properties of MDS codes, giving an additional advantage of efficient repair consuming minimum possible bandwidth. The repair bandwidth for a single failure is lower bounded by [6]:

$$\gamma_{MSR}^{min} \leq \frac{M}{k}\frac{n-1}{n-k}. \tag{1}$$

The length of the vector of symbols that a node stores in a single instance of the code determines the *sub-packetization level* α. The data from a failed node is recovered by transferring β symbols where $\beta < \alpha$ from each of d non-failed nodes called *helpers*. Thus, the repair bandwidth γ is equal to $d\beta$ where $\alpha \leq d\beta \ll M$. The repair bandwidth for a single failure is minimized when a fraction of $\frac{1}{r} = \frac{1}{n-k}$ of the stored data in all $d = n-1$ helpers is transmitted (Eq.1). MDS codes that achieve the minimum bandwidth are called *access-optimal codes*. However, for these codes, the access operations are in a non-contiguous manner, meaning that the number of I/Os can be several orders of magnitude greater than with RS codes. Working with a large sub-packetization level increases enormously the I/Os, and hence it has a negative impact on the speed of repair, the encoding throughput, the CPU utilization, the data availability etc. Consequently, implementing MSR codes in distributed storage systems can be quite complex and due to high number of I/Os can result in poor system performance.

MSR codes optimize only the bandwidth for a single failure. Although single failures are dominant [7], multiple node failures are often correlated and co-occurring in practice [8], [9]. Thus, we believe that for a successful and generally accepted practical implementation of erasure codes in distributed storage systems it is necessary to have MDS erasure codes that provide low repair bandwidth and low number of I/Os for arbitrary sub-packetization levels for single and multiple failures.

A. Our Contribution

In this paper, we study both the theoretical and the practical aspects of the explicit construction of general sub-packetized erasure codes that was recently introduced in [1]. Since the codes, upon their definition were presented without any specific name, for easier referencing we call them here *HashTag Erasure Codes (HTECs)* due to the resemblance between the hashtag sign # and the procedure of their construction. HTECs are simultaneously optimal in terms of storage, reliability and repair bandwidth and in this paper we study their I/O performance. We also study their recovery properties not just for single failures of the systematic nodes but also for multiple failures of the systematic nodes.

The first contribution of this paper is that we analyze many concrete instances of HTECs. HTEC construction is an explicit construction of MDS codes for an arbitrary sub-packetization level α. We show that the bandwidth savings for a single failure compared to RS codes [5] can be up to 60% and compared to Piggyback codes [10] can be up to 30%. We also show that for double failures, the codes still achieve repair bandwidth savings of 20% compared to RS codes. The code construction is general and works for any combination of n, k and α even when r is not a divisor of k. HTECs are access-optimal codes for $\alpha = r^{\left\lceil \frac{k}{r} \right\rceil}$. For all other values of α, HTECs achieve all points that lie from the MSR point to the

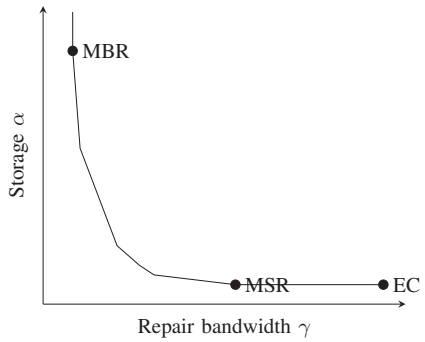


Fig. 1. Regenerating codes (MSR and MBR) offer performance improvement compared to conventional erasure coding (EC). We propose explicit constructions of MDS codes (codes that lie on the curve from MSR codes to EC).

RS erasure code (EC) point on the optimal trade-off curve between the storage and the repair bandwidth shown in Fig. 1.

Our second contribution is that we elaborate the correlation between the repair bandwidth and the I/Os in terms of α. While large values of α guarantee low average repair bandwidth, they inevitably increase the system I/Os (especially the number of random access operations) that has an impact on the repair speed, the overall latency of the recovery operation, the throughput, CPU utilization etc. That means that large values of α create I/O bottlenecks. From this perspective HTECs can be constructed for various values of α, hence they address one crucial practical engineering problem: how to identify the values of α that give optimal overall system performance. We show that all k systematic nodes are clustered in subsets of r nodes (with the exception of the last subset that can have less than r nodes). Then, we prove that there is one subset that can be repaired with $n-1$ random reads. Further on, for all other subsets of r nodes the discontiguity increases sequentially.

Our third contribution is a deeper scrutiny of the repair process with HTECs. In general, the repair process for a single failure is linear and highly parallel. This means a set of $\left\lceil \frac{\alpha}{r} \right\rceil$ symbols is independently repaired first and used along with the accessed data from other nodes to repair the remaining symbols of the failed node. We show that HTECs have one extra beneficial feature compared to RS codes: the amount of accessed and transferred data when multiple failures occur is less than RS codes. The main idea when recovering from multiple failures is to access and transfer the same data from the helpers as would have been done under a recovery from a single failure of the systematic nodes. To the best of our knowledge, HTECs are the first codes in the literature that offer bandwidth savings when recovering from multiple failures for any code parameters including the high-rate regime.

B. Paper Outline

The rest of the paper is organized as follows. Section II reviews the state-of-the-art for MDS erasure codes for distributed storage. Section III provides the mathematical preliminaries about HTECs and their properties. Section IV shows different code examples constructed with the algorithms from the previous Section. An algorithm for I/O optimization is presented in Section V. A comparison between the performance with HTECs and other codes in the literature is given in Section VI. Section VII discusses open issues, and finally, Section VIII concludes the paper.

II. Related Works

There has been a considerable amount of work in the area of erasure codes for efficient repair in distributed storage. We only review the most relevant literature on exact repair codes, since HTECs

TABLE I
COMPARISON OF EXPLICIT MDS CODES

Code construction	MDS	k, r supported	Sub-packetization level	Optimized for t failures
High-rate MSR [11]	Y	$r = \frac{k}{m}, m \geq 1$	$r^{\frac{k}{r}}$	$t = 1$
MSR over small fields [12]	Y	$r = 2, 3$	$r^{\frac{k}{r}}$	$t = 1$
Product-Matrix MSR [13]	Y	$r \geq k - 1$	r	$t = 1$
Piggyback 1 [10]	Y	All	$2m$, $m \geq 1$	$t = 1$
Piggyback 2 [10]	Y	$r \geq 3$	$(2r - 3)m$, $m \geq 1$	$t = 1$
Rotated RS [14]	Y	$r \in \{2, 3\}, k \leq 36$	2	$t = 1$
EVENODD, RDP [15], [16]	Y	$r = 2$	k	$t = 1$
MSCR [17]	Y	$r = k$	r	$2 \leq t \leq r$
CORE [18]	Y	$r = k$	r	$1 \leq t \leq r$
HashTag Erasure Codes (HTEC)	Y	All	$2 \leq \alpha \leq r^{\left\lceil \frac{k}{r} \right\rceil}$	$1 \leq t \leq r$

analyzed in this paper belong to the class of *exact repair codes* where the reconstructed data is exactly the same as the lost data.

In [19], high-rate $(n, k = n - 2, d = n - 1)$ MSR codes using Hadamard designs for optimal repair of both the systematic and parity nodes were constructed. The work also presented a general construction for optimal repair only of the systematic nodes for a sub-packetization level α of r^k. Codes for optimal systematic-repair for the same sub-packetization level appeared in [20] and [21]. The work was subsequently extended in [22] to include optimal repair of parity nodes.

Furthermore, the work in [23] showed that the required sub-packetization level α of access-optimal MSR codes for repair of any systematic node is $\alpha = r^{\frac{k}{r}}$. Few code constructions for optimal systematic-repair for $\alpha = r^{\frac{k}{r}}$ followed in the literature. In [24], Cadambe et al. proposed a high-rate MSR construction that is not explicit and requires a large field size. Later, an alternate construction of access-optimal MSR codes with a sub-packetization level of $\alpha = r^m$ where $m = \frac{k}{r}$ was presented in [11]. An essential condition for the code construction in [11] is that $m = \frac{k}{r}$ has to be an integer $m \geq 1$ where k is set to rm and α to r^m. Explicit access-optimal systematic-repair MSR codes over small finite fields for $\alpha \geq r^{\frac{k}{r}}$ were presented in [12]. However, these codes are limited for $r = 2, 3$.

Although the aforementioned constructions achieve the lower bound of the repair bandwidth for a single failure, as far as we know, they have not been practically implemented in real-world distributed storage systems. There are at least two practical reasons for that: either MSR codes require encoding/decoding operations over an exponentially growing finite field or the sub-packetization level α increases exponentially. Practical scenarios [25]–[27] showed that a good erasure code has to provide a satisfactory tradeoff between the system-level metrics such as storage overhead, reliability, repair bandwidth and I/Os. One way to achieve a good system tradeoff is to work with small sub-packetization levels.

Piggyback codes [10] are a good example of practical MDS codes with small sub-packetization levels. The basic idea of the piggyback framework is to take multiple instances of an existing code and add carefully designed functions of the data from one instance to another. Piggyback codes have better repair bandwidth performance than Rotated-RS [14], EVENODD [15] and RDP codes [16]. Rotated-RS codes exist only for $r \in \{2, 3\}$ and $k \leq 36$, while EVENODD and RDP codes exist only for $r = 2$. In [27], Rashmi et al. reported 35% bandwidth savings for an $(14, 10)$ code with $\alpha = 2$ when repairing a systematic node compared to an $(14, 10)$ RS code. The code construction [1] studied in this paper offers up to 67.5% bandwidth savings during repair of a single systematic node of an $(14, 10)$ code for α equal to $r^{\left\lceil \frac{k}{r} \right\rceil} = 64$.

Another way to provide good overall system performance is to optimize in terms of I/Os while still retaining the storage and bandwidth optimality. An algorithm that transforms Product-Matrix-MSR codes [13] into I/O optimal codes (termed PM-RBT codes) was presented in [28]. However, PM-RBT exist only for $r \geq k - 1$, i.e., the codes have low rates.

All MDS erasure codes discussed in the previous paragraphs are for optimal repair of a single node failure. Next we review codes that outperform MSR codes when multiple failures happen.

A cooperative recovery mechanism in the minimum-storage regime for repairing from multiple failures was proposed in [29], [30]. Minimum Storage Collaborative Regenerating (MSCR) codes minimize the repair bandwidth while still keeping the MDS property by allowing new nodes to download data from the non-failed nodes and the new nodes to exchange data among them. The repair bandwidth for MSCR codes under functional repair was derived independently in [29], [31]. The existence of a random linear strong-MDS code under the assumption that the operations are in a sufficiently large finite field was showed in [29]. The codes attain the MSR point but the decoding complexity is quite expensive. Adaptive regenerating codes where the numbers of failed and surviving nodes change over time were proposed in [31].

The authors in [32] showed that it is possible to construct exact MSCR codes for optimal repair of two failures directly from existing exact MSR codes. MSCR codes that cooperatively repair any number of systematic nodes and parity nodes or a combination of one systematic and one parity node were presented in [17]. However, the code rate of these codes is low ($n = 2k$). A study about the practical aspects of codes with the same code rate ($n = 2k$) in a system called CORE that supports multiple node failures can be found in [18]. There is no explicit construction of high-rate MDS codes for exact repair of multiple failures at the time of writing of this paper.

Table I compares different explicit codes designed for efficient repair, with respect to whether they are MDS or not, the parameters they support, the sub-packetization level and for how many failures the codes are optimized.

III. Mathematical preliminaries

Let us first give a list of major notations used in the paper:

n	total number of nodes
k	number of systematic nodes
r	number of parity nodes
d	number of non-failed nodes (helpers)
t	number of failures, $1 \leq t \leq r$
d_i	the i-th systematic node, $1 \leq i \leq k$
p_i	the i-th parity node, $1 \leq i \leq r$
M	size of the original data
α	sub-packetization level
β	data transferred from a node
γ	data accessed and transferred per node repair.

We consider only systematic codes where k nodes store the data without encoding it. We refer to these nodes as systematic nodes and the remaining $r = n - k$ nodes are called parity nodes.

MDS codes are optimal with respect to the storage-reliability tradeoff, because they offer maximum fault tolerance (r arbitrary failures) for the storage overhead consumed.

Dimakis et al. introduced the repair bandwidth as a new parameter of optimality for storage codes [6]. MDS codes that achieve the lower bound of the repair bandwidth are termed optimal with respect to the storage-bandwidth tradeoff. MSR codes are optimal with respect to the both storage-reliability and storage-bandwidth tradeoff.

In addition to the parameters n and k, MSR codes are associated with the parameters d (number of helpers) and α (sub-packetization level). In general, attaining the lower bound of the repair bandwidth requires an exponential increase of the finite field size that significantly increases the computational cost and the number of blocks α that the data is split in grows exponentially.

Accessing and transferring many small blocks of data increases enormously the degree of discontiguity and forfeits the potential disk I/O savings during data reconstruction.

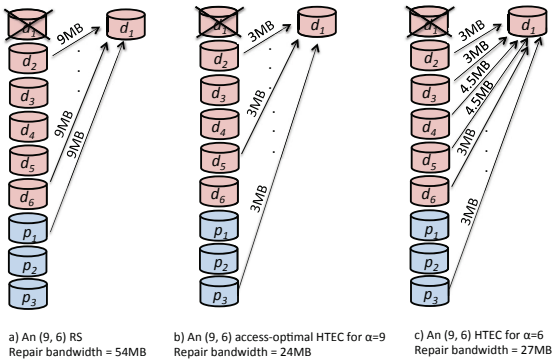


Fig. 2. Amount of accessed and transferred data for reconstruction of the systematic node d_1 for an (9,6) RS code, an (9, 6) access-optimal HTEC for $\alpha = 9$ and an (9, 6) HTEC for $\alpha = 6$. The systematic nodes are represented in red and the parity nodes in blue.

We have already presented some general properties about RS and MSR codes in Section I. We illustrate them with examples in the next two subsections and motivate the need for HTECs.

A. An Example with Reed-Solomon Codes

Let us consider the following example with a RS code for $k = 6$ and $r = 3$. The storage overhead is 50% and the code can recover from up to 3 failures. Fig. 2a depicts the storage of a file of 54MB across 9 nodes where each node stores 9MB. It also illustrates the reconstruction of node d_1 from nodes $d_2, \ldots, d_6$ and p_1. In order to reconstruct 9MB of unavailable data, 6×9MB=54MB are read from 6 nodes and transferred across the network to the node performing the decoding computations. The same amount of data (54MB) is needed to repair from up to 3 node failures. The number of random reads for this example is 6, because data is read in a contiguous manner from 6 different locations.

B. Two Examples with HashTag Erasure Codes

The first construction of HTECs appeared in [1] (although without their name HashTag Erasure Codes). They are MSR codes when $\alpha = r^{\left\lceil \frac{k}{r} \right\rceil}$ and $d = n - 1$. Hence, the generality of HTECs is that any code parameters are supported even when $\frac{k}{r}$ is a non-integer and α can be an arbitrary value. Note that such a flexible construction is not present in the proposals published in [11], [23], [24].

First we illustrate the performance improvement with an $(9, 6)$ access-optimal HTEC for $\alpha = 9$ and $d = 8$ compared to an $(9, 6)$ RS code. The bandwidth to repair any systematic node is reduced by 55.6% compared to an $(9, 6)$ RS code. The reconstruction of node d_1 is illustrated in Fig. 2b. In order to reconstruct the data from d_1, $1/3$ of the stored data from all 8 helpers is accessed and transferred, hence the repair bandwidth is only 24MB compared to 54MB with RS. HTECs achieve the minimum repair bandwidth given in Eq. 1 when $\alpha = 9$. Contacting 8 nodes when recovering a single failure increases the number of seek operations and random I/Os.

One way to reduce the number of random I/Os is to decrease α. The code presented in Fig. 2c is for $\alpha = 6$ where the reconstruction of d_1 is shown. Namely, 3MB are accessed and transferred from d_2, d_3, d_6, p_1, p_2 and p_3, while 4.5MB from d_4 and d_5. Thus, the repair bandwidth for d_1 is 27MB. The same amount of data is needed to repair d_3, d_4 and d_6, while 30MB of data is needed to repair d_2 and d_5. The average repair bandwidth, defined as the ratio of the total repair bandwidth to repair all systematic nodes to the file size, for the systematic nodes is 28MB. Implementing an erasure code with $\alpha = 6$ is simpler than with $\alpha = 9$ and it still provides bandwidth savings compared to 54MB with RS (and it is slightly more than 24MB with the MSR code for $\alpha = 9$). The big savings that come from the bandwidth reduction are evident when storing petabytes of data.

C. Definition of HashTag Erasure Codes

The general algorithm introduced in [1] offers a rich design space for constructing HTECs for various combinations of k systematic nodes, r parity nodes (so the total number of nodes is $n = k + r$), and sub-packetization levels α. In this subsection, we give a high level description of the algorithm for HTEC construction in Alg. 1.

As a general notation we say that a systematic node d_j, where $1 \leq j \leq k$, consists of α symbols $\{a_{1,j}, a_{2,j}, \ldots, a_{\alpha,j}\}$. The set $N = \{d_1, \ldots, d_k\}$ of k systematic nodes is partitioned in $\left\lceil \frac{k}{r} \right\rceil$ disjunctive subsets $J_1, J_2, \ldots, J_{\left\lceil \frac{k}{r} \right\rceil}$ where $|J_\nu| = r$ (if r does not divide k then $J_{\left\lceil \frac{k}{r} \right\rceil}$ has $k \mod r$ elements) and $N = \cup_{\nu=1}^{\left\lceil \frac{k}{r} \right\rceil} J_\nu$.

The basic idea for how to obtain the linear dependencies for the r parity nodes can be described as setting up r grids where the α symbols of each systematic node are represented as columns in that grid, and where the linear dependencies for the α symbols of the parity nodes are obtained from the rows of that grid. The name *HashTag Erasure Codes* comes from their resemblance with the popular hashtag sign #. In other words, the basic data structure component for construction of HTECs is an index array $P = ((i,j))_{\alpha \times k}$ of size $\alpha \times k$ where $\alpha \leq r^{\left\lceil \frac{k}{r} \right\rceil}$. The elements $p_{i,1}$, $i = 1, \ldots, \alpha$, in p_1 are a linear combination only of the symbols with indexes present in the rows of P_1. Actually, in Step 1 of the initialization phase r such arrays are constructed. Then, in Step 2 additional $\left\lceil \frac{k}{r} \right\rceil$ columns with pairs $(0,0)$ are added to $P_2, \ldots, P_r$.

In the next steps of Alg. 1, the zero pairs are replaced with concrete (i,j) pairs so that the repair bandwidth is minimized for a given sub-packetization level α. The values of α and k determine two phases of the algorithm. In the first phase, the indexes (i,j) that replace the $(0,0)$ pairs are chosen such that both Condition 1 and Condition 2 are satisfied (defined further in this section). The first phase starts with a granulation level parameter called run that is initialized to $\left\lceil \frac{\alpha}{r} \right\rceil$ and a parameter called $step$ initialized to 0. These parameters affect how the i indexes of the elements in d_j, $D = \{1, \ldots, \alpha\}$, are split into r disjunctive subsets $D_{d_j} = \cup_{\rho=1}^{r} D_{\rho,d_j}$. The granulation level decreases by a factor r with every round. Once the granulation level becomes equal to 1 and there are still $(0,0)$ pairs that have to get some (i,j) values from the unscheduled elements in the systematic nodes, the second phase starts where the remaining indexes are chosen such that only Condition 2 is satisfied.

- *Condition 1*: At least one subset D_{ρ,d_j} has $\left\lceil \frac{\alpha}{r} \right\rceil$ elements with runs of run consecutive elements separated with a distance between the indexes equal to $step$. The elements of that subset correspond to the row indexes in the $(k+\nu)$-th column, where $\nu = 1, \ldots, \left\lceil \frac{k}{r} \right\rceil$, in one of the arrays $P_2, \ldots, P_r$ that are all zero pairs. The distance between two elements in one node is computed in a cyclical manner such that the distance between the elements $a_{\alpha-1}$ and a_2 is 2.
- *Condition 2*: A necessary condition for the valid partitioning to achieve the lowest possible repair bandwidth is $D_{d_{j_1}} = D_{d_{j_2}}$ for all d_{j_1} and d_{j_2} in J_ν and $D_{\rho,d_{j_1}} \neq D_{\rho,d_{j_2}}$ for all d_{j_1} and d_{j_2} systematic nodes in the system. If $\left\lceil \frac{\alpha}{r} \right\rceil$ divides α, then D_{ρ,d_j} for all d_j in J_ν are disjunctive, i.e., $D = \cup_{j=1}^{r} D_{\rho,d_j} = \{1, \ldots, \alpha\}$.

Once the index arrays $P_1, \ldots, P_r$ are determined, the symbols $p_{i,l}$ in the parity nodes, $1 \leq i \leq \alpha$ and $1 \leq l \leq r$, are generated as a combination of the elements a_{j_1,j_2} where the pair (j_1, j_2) is in the i-th row of the index array P_l, i.e.,

$$p_{i,l} = \sum c_{l,i,j} a_{j_1,j_2}. \tag{2}$$

The linear relations have to guarantee an MDS code, i.e., to guarantee that the entire information can be recovered from any k nodes (systematic or parity).

Algorithm 1 High level description of an algorithm for generating HTEC for an arbitrary sub-packetization level

Input: n, k, α;

Output: Index arrays $P_1, \ldots, P_r$.

Initialization: $P_1, \ldots, P_r$ are initialized as index arrays $P = ((i,j))_{\alpha \times k}$;
Append $\left\lceil \frac{k}{r} \right\rceil$ columns to $P_2, \ldots, P_r$ all initialized to $(0,0)$;
Phase 1
Set the granulation level $run \leftarrow \left\lceil \frac{\alpha}{r} \right\rceil$ and $step \leftarrow 0$;
repeat
    Replace $(0,0)$ pairs with indexes (i,j) such that both Condition 1 and Condition 2 are satisfied;
    Decrease the granulation level run by a factor r and $step \leftarrow \left\lceil \frac{\alpha}{r} \right\rceil - run$;
until The granulation level $run > 1$
Phase 2
If there are still $(0,0)$ and unscheduled elements from the systematic nodes, choose (i,j) such that only Condition 2 is satisfied;
Return the index arrays $P_1, \ldots, P_r$.

D. MDS Property

Next we show that there always exists a set of non-zero coefficients from $\mathbf{F}_q$ in the linear combinations so that the code is MDS. We adapt Theorem 4.1 from [11] as follows:

Theorem 1: There exists a choice of non-zero coefficients $c_{l,i,j}$ where $l = 1, \ldots, r$, $i = 1, \ldots, \alpha$ and $j = 1, \ldots, k$ from $\mathbf{F}_q$ such that the code is MDS if $q \geq \binom{n}{k} r\alpha$.

Proof: The system of linear equations given in (2) defines a system of $r \times \alpha$ linear equations with $k \times \alpha$ variables. A repair of one failed node is given in Alg. 2, but for the sake of this proof, we explain the repair by discussing the solutions of the system of equations. When one node has failed, we have an overdefined system of $r \times \alpha$ linear equations with α unknowns. In general this can lead to a situation where there is no solution. However, since the values in system (2) are obtained from the values of the lost node, we know that there exists one solution. Thus, solving this system of $r \times \alpha$ linear equations with an overwhelming probability gives a unique solution, i.e., the lost node is recovered. When 2 nodes have failed, we have a system of $r \times \alpha$ linear equations with 2α unknowns. The same discussion for the overdefined system applies here. The most important case is when $r = n - k$ nodes have failed. In this case, we have a system of $r \times \alpha$ linear equations with $r \times \alpha$ unknowns. If the size of the finite field $\mathbf{F}_q$ is large enough, i.e., $q \geq \binom{n}{k} r\alpha$, as it is shown in Theorem 4.1 in [11], the system has a unique solution, i.e., the file M can be collected from any k nodes. ■

As a conclusion from Theorem 1 we say that HTECs as any other MDS codes are storage-reliability optimal.

Note that we do not address the problem of the size of the finite field. For the sake of the correctness of Theorem 1 we use the work in [11] where the lower bound for the size of the finite field is relatively big. On the other hand, in all examples in this paper we actually work with very small finite fields ($\mathbf{F}_{16}$ and $\mathbf{F}_{32}$). We leave the problem of determining the lower bound of the size of the finite field for HTECs as an open problem.

E. Repairing from a Single Systematic Failure

Alg. 2 shows how to repair a single systematic node where the systematic and the parity nodes are global variables. A set of $\left\lceil \frac{\alpha}{r} \right\rceil$ symbols is accessed and transferred from each of $n-1$ helpers. If $\alpha \neq r^{\left\lceil \frac{k}{r} \right\rceil}$, then additional elements may be required as described in Step 4. Note that a specific element is transferred just once and stored in a buffer. For every subsequent use of that element, the element is read from the buffer and further transfer operation is not required. The repair process is highly parallel, because a set

of $\left\lceil \frac{\alpha}{r} \right\rceil$ symbols is independently and in parallel repaired in Step 2 and then the remaining symbols are recovered in parallel in Step 5.

Algorithm 2 Repair of a systematic node d_l
Input: l;
Output: d_l.

1: Access and transfer $(k-1)\left\lceil \frac{\alpha}{r} \right\rceil$ elements $a_{i,j}$ from all $k-1$ non-failed systematic nodes and $\left\lceil \frac{\alpha}{r} \right\rceil$ elements $p_{i,1}$ from p_1 where $i \in D_{\rho,d_l}$;
2: Repair $a_{i,l}$ where $i \in D_{\rho,d_l}$
3: Access and transfer $(r-1)\left\lceil \frac{\alpha}{r} \right\rceil$ elements $p_{i,j}$ from $p_2, \ldots, p_r$ where $i \in D_{\rho,d_l}$;
4: Access and transfer from the systematic nodes the elements $a_{i,j}$ listed in the $i-$th row of the arrays $P_2, \ldots, P_r$ where $i \in D_{\rho,d_l}$ that have not been read in Step 1;
5: Repair $a_{i,l}$ where $i \in \mathcal{D} \setminus D_{\rho,d_l}$.

F. Repair Bandwidth

The bandwidth optimality of HTEC construction is captured in the following Proposition.

Proposition 1: If r divides α, then the indexes (i,j) of the elements $a_{i,j}$ where $i \in \mathcal{D} \setminus D_{\rho,d_j}$ for each group of r systematic nodes are scheduled in one of the $\left\lceil \frac{k}{r} \right\rceil$ additional columns in the index arrays $P_2, \ldots, P_r$.

Proof: The proof is a simple counting strategy of all indexes (i,j) of the elements $a_{i,j}$ where $i \in \mathcal{D} \setminus D_{\rho,d_j}$. ■

Proposition 2: The bandwidth for repair of a single systematic node is bounded between the following lower and upper bounds:

$$\frac{(n-1)}{r} \leq \gamma \leq \frac{(n-1)}{r} + \frac{(r-1)}{\alpha}\left\lceil \frac{\alpha}{r} \right\rceil \left\lceil \frac{k}{r} \right\rceil. \tag{3}$$

Proof: Note that we read in total $k\left\lceil \frac{\alpha}{r} \right\rceil$ elements in Step 1 of Alg. 2. Additionally, $(r-1)\left\lceil \frac{\alpha}{r} \right\rceil$ elements are read in Step 3. Assuming that we do not read more elements in Step 4 and every element has a size of $\frac{1}{\alpha}$, we determine the lower bound as $\frac{(n-1)}{r}$. This bound is the same as the one given in Eq. 1. To derive the upper bound, we assume that we read all elements $a_{i,j}$ from the extra $\left\lceil \frac{k}{r} \right\rceil$ columns of the arrays $P_2, \ldots, P_r$ in Step 4. Thus, the upper bound is $\frac{(n-1)}{r} + \frac{(r-1)}{\alpha}\left\lceil \frac{\alpha}{r} \right\rceil \left\lceil \frac{k}{r} \right\rceil$. ■

Proposition 3: The recovery bandwidth is equal for all systematic nodes when k is a divisor of $\alpha = r\left\lceil \frac{k}{r} \right\rceil$.

Proof: If k divides α then the steps in Alg. 1 produce index arrays $P_1, \ldots, P_r$ where the distribution of indexes from all systematic nodes is completely symmetric (or balanced). That symmetry reflects to the linear dependencies for each of the parity elements which further implies that the recovery bandwidth is symmetrical i.e., equal for all systematic nodes. ■

G. Repairing from Multiple Systematic Failures

Alg. 3 shows how to find a minimal system of linear equations to repair t failures where $1 \leq t \leq r$ with minimal bandwidth. Data from all $n-t$ non-failed nodes is accessed and transferred. The sets N and T consist of the indexes of all systematic nodes and the failed systematic nodes, respectively. Note that when $t = 1$ the amount of accessed and transferred data is the same with both Alg. 2 and Alg. 3.

Algorithm 3 Repair of t systematic nodes where $1 \leq t \leq r$
Input: $T = \{l_1, \ldots, l_t\}$ where $T \subset N$ and $|T| = t$;
Output: Data from all d_l where $l \in T$.

1: **for** each $l \in T$ **do**
2: Select equations $p_{i,l}$ where $i \in D_{\rho,d_l}$ from $p_1, \ldots, p_r$;
3: **end for**
4: **while** The set of selected equations does not contain all missing $t \times \alpha$ elements $a_{i,l}$ where $i = 1, \ldots, \alpha$ and $l \in T$ **do**
5: Select equation $p_{i,j}$ where $i \in \mathcal{D} \setminus \cup_{j=l_1}^{l_t} D_{\rho,d_j}$ that includes maximum number of new non-included elements $a_{i,l}$;
6: **end while**
7: Access and transfer from the available systematic nodes and from the parity nodes all elements $a_{i,j}$ and $p_{i,j}$ listed in the selected equations;
8: Solve the system for $t \times \alpha$ unknowns $a_{i,l}$ where $i = 1, \ldots, \alpha$ and $l \in T$;
9: Return the data for the missing d_l where $l \in T$.

H. Repair Bandwidth

Proposition 4: The bandwidth to repair t systematic nodes is bounded between the following lower and upper bounds:

$$\frac{t}{\alpha}\left\lceil\frac{\alpha}{r}\right\rceil(n-t) \leq \gamma \leq k\alpha. \tag{4}$$

Proof: Note that if for all missing nodes d_l where $l \in T$ it is true that the index sets D_{ρ,d_l} are disjunctive, i.e., it is true that $D_{\rho,d_{l_1}} \cap D_{\rho,d_{l_2}} = \emptyset$ where $l_1, l_2 \in T$, then in Steps 1 – 3 of Alg. 3 we will select all $t \times \alpha$ necessary equations to repair the t missing nodes. In that case Alg. 3 selects the minimum number of linear equations and thus the repair bandwidth reaches the lower bound. This means that in the Step 8 we need to read in total $t(k-t)\left\lceil\frac{\alpha}{r}\right\rceil$ elements $a_{i,j}$ from the $k-t$ systematic nodes and additionally to read $t \cdot r\left\lceil\frac{\alpha}{r}\right\rceil$ elements $p_{i,j}$ from the r parity nodes. Assuming that every element has a size of $\frac{1}{\alpha}$, we determine the lower bound as $\frac{t}{\alpha}((k-t)\left\lceil\frac{\alpha}{r}\right\rceil + r\left\lceil\frac{\alpha}{r}\right\rceil) = \frac{t}{\alpha}\left\lceil\frac{\alpha}{r}\right\rceil(n-t)$.

Any additional selection of equations in the while loop in Steps 4 – 6 just increases the repair bandwidth and can not exceed the upper bound which is simply the same amount of repair bandwidth required for RS codes that is $k\alpha$. ■

IV. Code Examples with Arbitrary Sub-packetization Levels and Multiple Failures

In this Section, we give two examples for an $(9, 6)$ HTEC code for two sub-packetization levels: $\alpha = 6$ and $\alpha = 9$. We choose the code $(9, 6)$ due to its relatively small size that is appropriate for presentation.

A. An $(9, 6)$ HTEC for $\alpha = 6$

The systematic nodes $d_1, \ldots, d_6$ and the parity nodes p_1, p_2, p_3 are shown in Fig. 3. The file size M is 36 symbols where each node stores $\alpha = 6$ symbols. The elements of p_1 are linear combinations of the row elements from the systematic nodes multiplied with coefficients from $\mathbf{F_{16}}$. The elements of p_2 and p_3 are obtained by adding extra symbols to the row sum. We next show the code construction following the steps in Alg. 1.

1) Initialize P_i, $i = 1, 2, 3$, as index arrays $P_i = ((i, j))_{6\times 6}$,

$$P_i = \begin{bmatrix} (1,1) & (1,2) & (1,3) & (1,4) & (1,5) & (1,6) \\ (2,1) & (2,2) & (2,3) & (2,4) & (2,5) & (2,6) \\ (3,1) & (3,2) & (3,3) & (3,4) & (3,5) & (3,6) \\ (4,1) & (4,2) & (4,3) & (4,4) & (4,5) & (4,6) \\ (5,1) & (5,2) & (5,3) & (5,4) & (5,5) & (5,6) \\ (6,1) & (6,2) & (6,3) & (6,4) & (6,5) & (6,6) \end{bmatrix}.$$

2) Append $\left\lceil \frac{k}{r} \right\rceil = 2$ columns to P_2 and P_3 initialized to $(0,0)$, i.e., $P_2 = P_3 =$

$$\begin{bmatrix} (1,1) & (1,2) & (1,3) & (1,4) & (1,5) & (1,6) & (0,0) & (0,0) \\ (2,1) & (2,2) & (2,3) & (2,4) & (2,5) & (2,6) & (0,0) & (0,0) \\ (3,1) & (3,2) & (3,3) & (3,4) & (3,5) & (3,6) & (0,0) & (0,0) \\ (4,1) & (4,2) & (4,3) & (4,4) & (4,5) & (4,6) & (0,0) & (0,0) \\ (5,1) & (5,2) & (5,3) & (5,4) & (5,5) & (5,6) & (0,0) & (0,0) \\ (6,1) & (6,2) & (6,3) & (6,4) & (6,5) & (6,6) & (0,0) & (0,0) \end{bmatrix}.$$

3) For the systematic nodes d_1, d_2 and d_3 in J_1, $run = 2$ and $step = 0$.
4) Both Condition 1 and Condition 2 have to be fulfilled. We use the notation $D_{\rho,d_j}, j = 1, \ldots, 6$, to denote the subset with its elements corresponding to row indexes in the $(6+\nu)$-th column, $\nu = 1, 2$, in one of the arrays P_2 and P_3 that are all zero values $(0,0)$. For the node d_1, we get $D_{d_1} = \{\{1,2\},\{3,4\},\{5,6\}\}$. Note that the run length is 2 and the distance between the indexes is 0. The first 2 zero elements in the 7-th column of P_2 are at the positions $(1,7)$ and $(2,7)$, thus, $D_{\rho,d_1} = \{1,2\}$. The i indexes of the remaining pairs $(i,1)$ where $i = 4, \ldots, 6$ belong to 2 other subsets $D \setminus D_{\rho,d_1}$, i.e., $D_{2,d_1} = \{3,4\}$ and $D_{3,d_1} = \{5,6\}$. The pairs $(i,1)$ for $i \in D \setminus D_{\rho,d_1}$ are added in the 7-th column of P_2 and P_3.
Similarly, we perform the same steps for the nodes d_2 and d_3 resulting in $D_{\rho,d_2} = \{3,4\}$ and $D_{\rho,d_3} = \{5,6\}$, respectively.
Next we schedule the elements from d_4, d_5 and d_6.
5) For the nodes d_4, d_5 and d_6 in J_2, $run = 1$ and $step = 1$.
6) We perform the same steps as for the nodes in J_1. Here we only give the corresponding D_{ρ,d_j}, i.e., $D_{\rho,d_4} = \{1,3\}$, $D_{\rho,d_5} = \{2,5\}$ and $D_{\rho,d_6} = \{4,6\}$.
7) After replacing the $(0,0)$ pairs with specific (i,j) pairs, the final index arrays are:

$$P_1 = \begin{bmatrix} (1,1) & (1,2) & (1,3) & (1,4) & (1,5) & (1,6) \\ (2,1) & (2,2) & (2,3) & (2,4) & (2,5) & (2,6) \\ (3,1) & (3,2) & (3,3) & (3,4) & (3,5) & (3,6) \\ (4,1) & (4,2) & (4,3) & (4,4) & (4,5) & (4,6) \\ (5,1) & (5,2) & (5,3) & (5,4) & (5,5) & (5,6) \\ (6,1) & (6,2) & (6,3) & (6,4) & (6,5) & (6,6) \end{bmatrix},$$

$$P_2 = \begin{bmatrix} (1,1) & (1,2) & (1,3) & (1,4) & (1,5) & (1,6) & (3,1) & (2,4) \\ (2,1) & (2,2) & (2,3) & (2,4) & (2,5) & (2,6) & (4,1) & (1,5) \\ (3,1) & (3,2) & (3,3) & (3,4) & (3,5) & (3,6) & (1,2) & (5,4) \\ (4,1) & (4,2) & (4,3) & (4,4) & (4,5) & (4,6) & (2,2) & (1,6) \\ (5,1) & (5,2) & (5,3) & (5,4) & (5,5) & (5,6) & (1,3) & (3,5) \\ (6,1) & (6,2) & (6,3) & (6,4) & (6,5) & (6,6) & (2,3) & (3,6) \end{bmatrix}.$$

and

$$P_3 = \begin{bmatrix} (1,1) & (1,2) & (1,3) & (1,4) & (1,5) & (1,6) & (5,1) & (4,4) \\ (2,1) & (2,2) & (2,3) & (2,4) & (2,5) & (2,6) & (6,1) & (4,5) \\ (3,1) & (3,2) & (3,3) & (3,4) & (3,5) & (3,6) & (5,2) & (6,4) \\ (4,1) & (4,2) & (4,3) & (4,4) & (4,5) & (4,6) & (6,2) & (2,6) \\ (5,1) & (5,2) & (5,3) & (5,4) & (5,5) & (5,6) & (3,3) & (6,5) \\ (6,1) & (6,2) & (6,3) & (6,4) & (6,5) & (6,6) & (4,3) & (5,6) \end{bmatrix}.$$

8) Schedule the elements $a_{i,j}$ with the corresponding (i,j) indexes in the index arrays. The parity symbols are linear combinations from the elements in the same row in the array. The coefficients for the MDS code with 6 systematic and 3 parity nodes for $\alpha = 6$ in Fig. 3 are from $\mathbf{F}_{16}$ with irreducible polynomial $x^4 + x^3 + 1$.

We next show how to repair the node d_1 following Alg. 2. First, we repair the elements $a_{1,1}, a_{2,1}$. Thus, we access and transfer $a_{1,j}$ and $a_{2,j}$ where $j = 2, \ldots, 6$ from all 5 non-failed systematic nodes and $p_{1,1}$, $p_{2,1}$ from p_1. Since $a_{3,1}, a_{4,1}$ are added as extra elements in p_2, we need to access and transfer $p_{1,2}$ and $p_{2,2}$ from p_2. Due to the optimal scheduling of the extra elements in the parity nodes, no further elements are required to recover $a_{3,1}, a_{4,1}$. The last two elements $a_{5,1}, a_{6,1}$ are recovered by accessing and transferring $p_{1,3}, p_{2,3}, a_{4,4}$ and $a_{4,5}$. The last two elements are read because we work with a sub-packetization level that is not equal to 9. The data from d_1 is recovered by accessing and transferring in total 18 elements from 8 helpers. Exactly the same amount of data, 18 symbols, is needed to repair d_3, d_4 or d_6, while 20 symbols are needed to repair d_2 and d_5. Thus, the average repair bandwidth is equal to 3.11 symbols.

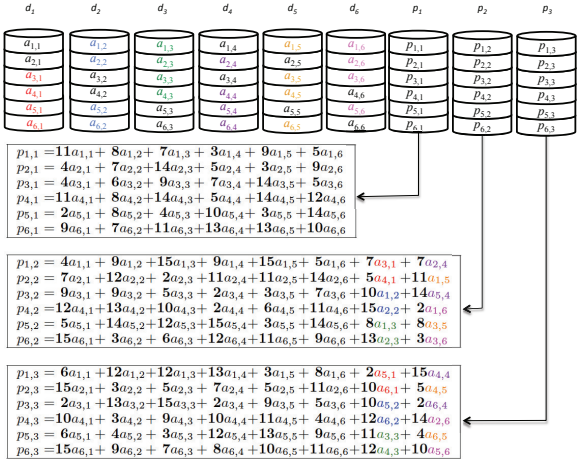


Fig. 3. An MDS array code with 6 systematic and 3 parity nodes for $\alpha = 6$. The elements presented in colors are scheduled as additional elements in p_2 and p_3. The coefficients are from $\mathbf{F}_{16}$ with irreducible polynomial $x^4 + x^3 + 1$.

B. An $(9, 6)$ HTEC for $\alpha = 9$ and Repairing from Multiple Failures

We next show the recovery of the nodes d_1 and d_3 under an $(9, 6)$ code for $\alpha = 9$ by using Alg. 3. The $(9, 6)$ code in Fig. 4 is generated by Alg. 1. Here we only give $D_{\rho,d_1} = \{1, 2, 3\}$ and $D_{\rho,d_3} = \{7, 8, 9\}$. We first access and transfer 24 $a_{i,j}$ elements from all 4 non-failed systematic nodes and 18 $p_{i,j}$ elements from p_1, p_2, p_3 where $i \in D_{\rho,d_1} \cup D_{\rho,d_3}$. In total we have accessed and transferred 42 symbols. We next check the condition if the number of linearly independent equations is equal to 18. When d_1 and d_3 are lost, the condition is fulfilled so it is possible to repair the 18 lost symbols from d_1 and d_3. Exactly the same amount of data, 42 symbols, is needed to repair any pair of lost systematic nodes when $D_{\rho,d_{l_i}} \cap D_{\rho,d_{l_j}} = \emptyset$. There are in total $\binom{3}{2}$ combinations of 2 failed systematic nodes from the nodes d_1, d_2 and d_3 in J_1 and $\binom{3}{2}$ combinations of 2 failed systematic nodes from the nodes d_4, d_5 and d_6 in J_2.

The recovery process has some additional steps when $D_{\rho,d_{l_i}} \cap D_{\rho,d_{l_j}} \neq \emptyset$, i.e., when 1 node from each of the groups J_1 and J_2 has failed. By reading the elements from Step 2 the number of linearly independent equations $p_{i,j}$ for $i \in D_{\rho,d_{l_i}} \cup D_{\rho,d_{l_j}}$ is exhausted. Thus, we have to read $p_{i,j}$ that has not been read previously, i.e., $p_{i,j}$ where $i \in \mathcal{D} \setminus D_{\rho,d_{l_i}} \cup D_{\rho,d_{l_j}}$. Let us consider the repair of d_1 and d_4 where $D_{\rho,d_1} = \{1, 2, 3\}$ and $D_{\rho,d_4} = \{1, 4, 7\}$. We first access and transfer 20 elements $a_{i,j}$ from all 4 non-failed systematic nodes and 15 $p_{i,j}$ elements from p_1, p_2, p_3 where $i \in D_{\rho,d_1} \cup D_{\rho,d_4}$. In total we have accessed and transferred 35 symbols. We next check if the number of linearly independent equations is equal to 18. Since we have only transferred 15 $p_{i,j}$, the condition is not fulfilled. So we need to read 3 more $p_{i,j}$ than have not been read previously. In this case, we transfer $p_{5,1}, p_{5,2}, p_{6,1}$ and the $a_{i,j}$ elements from the 5-th row in the parity arrays P_1 and P_2 and from the 6-th row in the parity array P_1 that have not been transferred in Step 2. The total number of symbols read to repair d_1 and d_4 is 46. There are in total $\binom{3}{1}\binom{3}{1}$ pairs of failed nodes where 46 symbols are needed to repair from double failures.

The average repair bandwidth to repair any 2 failed nodes is 4.933 symbols that is 17.783% reduction compared to a RS code.

V. Optimizing I/O During Repair

Optimizing the amount of data accessed and transferred might not directly correspond to an optimized I/O, unless the data reads are sequential. Motivated by the practical importance of I/O, we optimize the

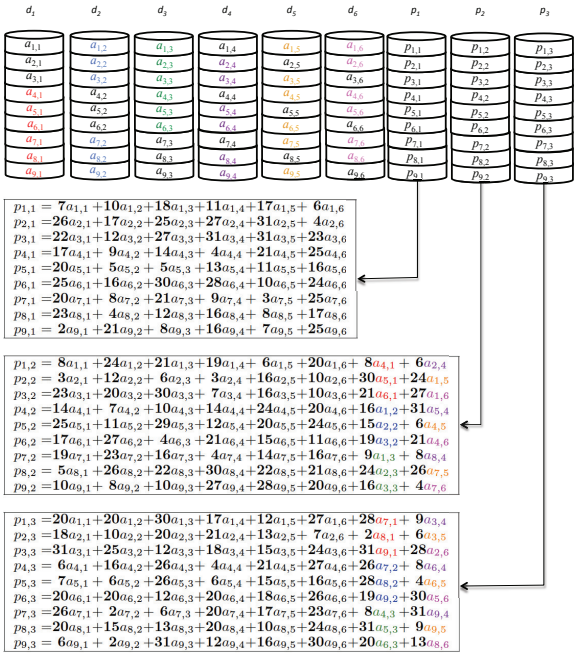


Fig. 4. An MDS array code with 6 systematic and 3 parity nodes for $\alpha = 9$. The elements presented in colors are scheduled as additional elements in p_2 and p_3. The coefficients are from $\mathbf{F}_{32}$ with irreducible polynomial $x^5 + x^3 + 1$.

MDS HashTag erasure codes constructed with Alg. 1 while still retaining their optimality in terms of storage and repair bandwidth.

Before discussing sequential and random reads, first we explain what do we treat as a sequential and as a random read. Since the amount of data-read and the amount of data-transferred for HTECs is equal, the number of read and transfer operations is the same. Hence we use the terms reads and transfers interchangeably. Whenever there is a seek for data from a new location, the first read is counted as a random read. If the data is read in a contiguous manner, then the second read is counted as a sequential read. For instance, when a seek request is initiated for $a_{1,1}$ from d_1 in Fig. 5, then the number of random reads is 1. If we next read $a_{2,1}, a_{3,1}$ and so forth in a contiguous manner, then the number of sequential reads increases by one for each contiguous access. On the other hand, if we read $a_{3,1}$ after reading $a_{1,1}$ (but not $a_{2,1}$), then the number of random reads becomes 2. Note that reading $a_{6,1}$ and $a_{1,1}$ results in 2 random reads.

The parameter $step$ defines the contiguity of the reads for the codes obtained by Alg. 1.

Theorem 2: The number of random reads for access-optimal codes is equal to $n-1$ for r out of the k systematic nodes.

Proof: When repairing a single systematic node when $\alpha = r\left\lceil \frac{k}{r} \right\rceil$, then data from all $n-1$ helpers has to be accessed and transferred. The set of systematic nodes N is partitioned in $\left\lceil \frac{k}{r} \right\rceil$ disjunctive subsets of r nodes (the last subset may have less than r nodes). For the group of r nodes in J_1, $step$ is equal to 0 and hence the reads are sequential. This means that in total $n-1$ seeks to read the data in a contiguous manner from $n-1$ helpers are performed. ■

The scheduling of indexes by Alg. 1 ensures a gradual increase in the number of random reads, hence no additional algorithms such as hop-and-couple [28] are needed to make the reads sequential, instead we

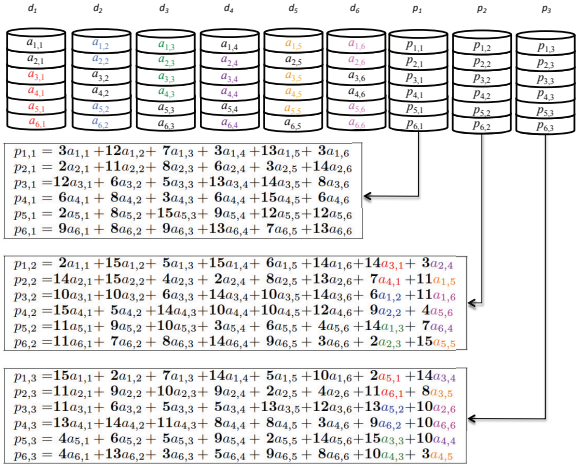


Fig. 5. An I/O optimized MDS array code with 6 systematic and 3 parity nodes for $\alpha = 6$. The elements presented in colors are scheduled as additional elements in p_2 and p_3. The coefficients are from $\mathbf{F}_{16}$ with irreducible polynomial $x^4 + x^3 + 1$.

use Alg. 4.

Algorithm 4 I/O optimization of an (n, k) code
Input: An (n, k) code generated with Alg. 1;
Output: An (n, k) I/O optimized code.

1: Find an (n, k) MDS erasure code where the parity nodes are generated with Alg. 1;
2: Repeatedly improve the solution by searching for codes with low I/O for single systematic node failure with same repair bandwidth, until no more improvements are necessary/possible.

A. Code Example: Optimizing I/O

Let us consider that the file size is 54MB and each node stores 9MB. Each I/O reads and transfers 512KB. When repairing a failed systematic node with an $(9, 6)$ RS code, 6 out of the 8 non-failed nodes have to be accessed (in total 6 random reads to recover 1 node). Since each I/O transfers 512KB and with the RS code the whole data of 9MB stored in a node is read and transferred, then there are 18 I/O transfers of 512KB from each node where the first I/O is random and the next 17 are sequential. Thus, the number of 512KB sized I/Os is 108 where 6 are random and 6×17=102 are sequential.

Next we revisit the first example presented in Section IV where $\alpha = 6$. In this example, we are working with blocks of 9MB/6=1.5MB. The average number of random reads for recovery of one systematic node is 13.33. Since the block size is 1.5MB and each random I/O transfers 512KB, that means that each random read is accompanied with 2 sequential reads. In addition, there are 5.33 sequential reads of blocks that becomes 5.33×3 reads of 512KB. In average there are 13.33 random I/Os and 42.66 sequential I/Os when reconstructing a lost systematic node, for a total of 56 I/Os.

With the help of Alg. 4, we reduce the number of random reads while still providing the same average repair bandwidth. The code given in Fig. 5 is a good example of an $(9, 6)$ I/O optimized code for $\alpha = 6$. For this code construction the number of random I/Os is reduced to 11.33, while the number of sequential I/Os becomes 44.66. This is achieved by using different D_{d_j} for the nodes $d_j \in J_2$. Namely, we obtained the following sets $D_{\rho,d_4} = \{1, 5\}$, $D_{\rho,d_5} = \{2, 6\}$ and $D_{\rho,d_6} = \{3, 4\}$. For instance, the ratio between the number of random I/Os and total number of I/Os for the non-optimized code is 0.238, while for the

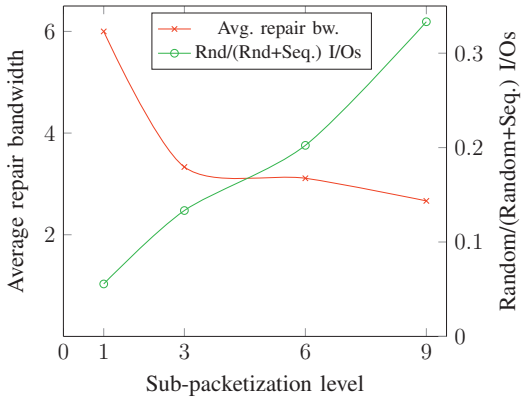


Fig. 6. Average repair bandwidth and normalized number of random I/Os for recovery of the systematic nodes for an $(9, 6)$ code for different sub-packetization levels.

optimized it is 0.202. This is an important improvement, since in practice the random reads are more expensive compared to the sequential ones.

Fig. 6 shows the relation between the average repair bandwidth (we consider that $\frac{M}{k} = 1$) for a single systematic failure and the normalized number of random reads with the sub-packetization level for an (9, 6) code. We observe that for RS code where $\alpha = 1$, the average repair bandwidth is biggest (the highest point on the red line with the value 6), while the randomness in the I/Os is the lowest. The repair bandwidth decreases as α increases and the minimum bandwidth of 2.67 is achieved for $\alpha = 9$. The situation is completely opposite when the metric of interest is the number of random reads. The number of reads (especially random reads) increases rapidly with α as shown in Fig. 6. The best overall system performance is achieved for α in the range between 3 and 6.

VI. Comparison of Different Codes

We have implemented HTECs in C/C++ and we next present performance comparison with other codes. We first compare the average data read and downloaded during a single node repair for different code parameters under HTEC construction and Piggyback constructions for systematic repair [10]. Fig. 7 shows a comparison between the data read and transferred when repairing a single systematic node with Piggyback 1, Piggyback 2 and HTECs. The plot corresponds to the sub-packetization level being 8 in Piggyback 1 and HTEC and $4(2r - 3)$ in Piggyback 2. We observe that HTEC construction requires less data read and less data transfer compared to Piggyback 1 and Piggyback 2 even though the sub-packetization level is lower than the one in Piggyback 2.

Fig. 8 shows the relation between the average repair bandwidth (we consider that $\frac{M}{k} = 1$) for a single failure and the sub-packetization level for an (14, 10) code. For $\alpha = 1$, we have the conventional Reed-Solomon codes and the average repair bandwidth is equal to k (the highest point on the red line with the value 10). A Hitchhiker code [27] for $\alpha = 2$ reduces the repair bandwidth by 35% compared to the RS code. The remaining values of the average repair bandwidth are for HTECs. We observe that the lowest repair bandwidth that is 3.25 is achieved for $\alpha = r^{\left\lceil \frac{k}{r} \right\rceil} = 64$. On the other hand, the highest number of reads is for $\alpha = 64$. Typically, an engineering decision would end up by choosing parameters for optimal overall system performance with α in the range between 4 and 16.

VII. Discussion

In this Section, we discuss some open issues that are not covered in this paper.

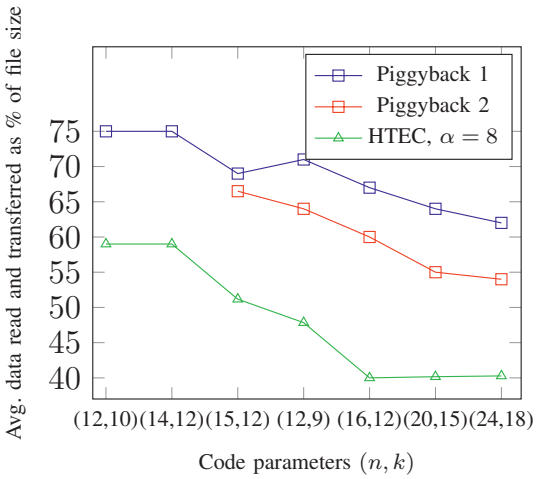


Fig. 7. Average data read and transferred for repair of a single systematic node in Piggyback 1 for $\alpha = 8$, Piggyback 2 for $\alpha = 4(2r - 3)$ and HTEC for $\alpha = 8$.

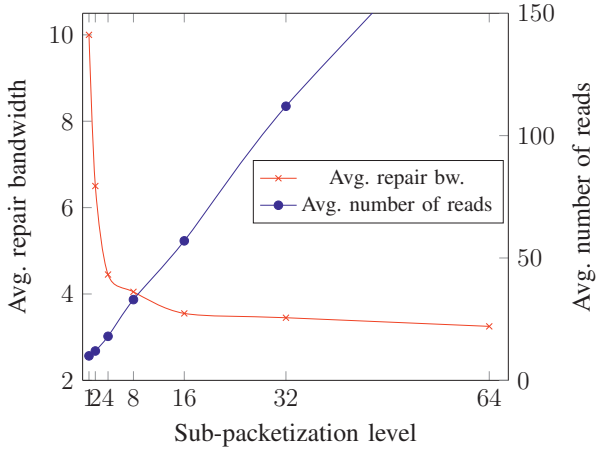


Fig. 8. Average repair bandwidth for the systematic nodes and average number of reads (sequential+random reads) for recovery of the systematic nodes for an $(14, 10)$ code for different sub-packetization levels.

Lower bound of the finite field size. In this paper, we use the work in [11] to guarantee the existence of non-zero coefficients from $\mathbf{F}_q$ so that the code is MDS. However, the lower bound of the size of the finite field is relatively big. On the other hand, in all examples in this paper we actually work with very small finite fields ($\mathbf{F}_{16}$ and $\mathbf{F}_{32}$). Recent results in [33] showed that a code is access-optimal for $\alpha = r^{\left\lceil \frac{k}{r} \right\rceil}$ over any finite field $\mathbf{F}$ as long as $|\mathbf{F}| \geq r\left\lceil \frac{k}{r} \right\rceil$. Determining the lower bound of the size of the finite field for HTECs remains an open problem.

Efficient repair of the parity nodes. HTEC construction considers only an efficient repair of the systematic nodes. Several high-rate MSR codes for efficient repair of both systematic and parity nodes [22], [34], [35] exist in the litetruare. Still for these codes, either the sub-packetization level is too large or

the constructions are not explicit. An open issue is how to extend the HTEC construction to support an efficient repair of the parity nodes as well.

VIII. Conclusions

MSR codes have been proposed as a superior alternative to popular RS codes in terms of storage, fault-tolerance and repair bandwidth. In this paper, we presented *HashTag Erasure Code (HTEC)* construction that provides the flexibility of constructing MDS codes for any code parameters including an arbitrary sub-packetization level. MSR codes are constructed when the sub-packetization level of HTECs is equal to $r^{\left\lceil \frac{k}{r} \right\rceil}$. Then HTECs are access-optimal and $n-1$ helpers are contacted during the repair process.

Existing MDS erasure code constructions for storage systems do not address the critical problem of I/O optimization. In this work, we show that HTECs can minimize the disk I/O consumed while simultaneously providing optimality in terms of storage, reliability and repair-bandwidth. We identify the range of the sub-packetization level that gives the best overall system performance. Additionally, we show that HTECs reduce the repair bandwidth for more than one failure. HTECs are the first high-rate MDS codes that tackle the problem of multiple failures.

References

[1] K. Kralevska, D. Gligoroski, and H. Øverby, "General sub-packetized access-optimal regenerating codes," *IEEE Communications Letters*, vol. 20, no. 7, pp. 1281–1284, July 2016.

[2] H. Weatherspoon and J. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *Proc. 1st Int. Workshop on Peer-to-Peer Systems*, 2002, pp. 328–338.

[3] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, ser. SOSP '03. ACM, 2003, pp. 29–43.

[4] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, ser. MSST '10. IEEE Computer Society, 2010, pp. 1–10.

[5] G. S. I. S. Reed, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[6] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sept. 2010.

[7] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the facebook warehouse cluster," in *5th USENIX Workshop on Hot Topics in Storage and File Systems*. USENIX, 2013.

[8] D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in globally distributed storage systems," in *Presented as part of the 9th USENIX Symposium on Operating Systems Design and Implementation*. USENIX, 2010.

[9] Y. Hu, H. C. H. Chen, P. P. C. Lee, and Y. Tang, "NCCloud: applying network coding for the storage repair in a cloud-of-clouds," in *FAST*. USENIX Association, 2012.

[10] K. V. Rashmi, N. B. Shah, and K. Ramchandran, "A piggybacking design framework for read-and download-efficient distributed storage codes," in *IEEE International Symposium on Information Theory Proceedings (ISIT)*, July 2013, pp. 331–335.

[11] G. K. Agarwal, B. Sasidharan, and P. V. Kumar, "An alternate construction of an access-optimal regenerating code with optimal sub-packetization level," in *Proc. 21st Nat. Conf. Comm.*, 2015, pp. 1–6.

[12] N. Raviv, N. Silberstein, and T. Etzion, "Access-optimal MSR codes with optimal sub-packetization over small fields," *CoRR*, vol. abs/1505.00919, 2015.

[13] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the msr and mbr points via a product-matrix construction," *IEEE Transactions on Information Theory*, vol. 57, no. 8, pp. 5227–5239, Aug 2011.

[14] O. Khan, R. C. Burns, J. S. Plank, W. Pierce, and C. Huang, "Rethinking erasure codes for cloud file systems: minimizing I/O for recovery and degraded reads," in *FAST*. USENIX Association, 2012, p. 20.

[15] M. Blaum, J. Brady, J. Bruck, and J. Menon, "Evenodd: an efficient scheme for tolerating double disk failures in raid architectures," *IEEE Transactions on Computers*, vol. 44, no. 2, pp. 192–202, Feb 1995.

[16] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar, "Row-diagonal parity for double disk failure correction," in *Proceedings of the USENIX FAST '04 Conference on File and Storage Technologies*. USENIX Association, Mar. 2004, pp. 1–14.

[17] J. Chen and K. W. Shum, "Repairing multiple failures in the suh-ramchandran regenerating codes," in *ISIT*. IEEE, 2013, pp. 1441–1445.

[18] R. Li, J. Lin, and P. P. C. Lee, "Enabling concurrent failure recovery for regenerating-coding-based storage systems: From theory to practice," *IEEE Transactions on Computers*, vol. 64, no. 7, pp. 1898–1911, July 2015.

[19] D. S. Papailiopoulos, A. G. Dimakis, and V. R. Cadambe, "Repair optimal erasure codes through hadamard designs," *IEEE Transactions on Information Theory*, vol. 59, no. 5, pp. 3021–3037, May 2013.

[20] V. R. Cadambe, C. Huang, and J. Li, "Permutation code: Optimal exact-repair of a single failed node in mds code based distributed storage systems," in *IEEE International Symposium on Information Theory Proceedings (ISIT)*, July 2011, pp. 1225–1229.
[21] I. Tamo, Z. Wang, and J. Bruck, "Zigzag codes: Mds array codes with optimal rebuilding," *IEEE Trans. Inf. Theory*, vol. 59, no. 3, pp. 1597–1616, March 2013.
[22] Z. Wang, I. Tamo, and J. Bruck, "On codes for optimal rebuilding access," in *Proc. 49th Annual Allerton Conf. Comm., Control, Comp.,*, 2011, pp. 1374–1381.
[23] I. Tamo, Z. Wang, and J. Bruc, "Access vs. bandwidth in codes for storage," in *Proc. IEEE Int. Symp. Inf. Theory*, 2012, pp. 1187–1191.
[24] V. R. Cadambe, C. Huang, J. Li, and S. Mehrotra, "Polynomial length mds codes with optimal repair in distributed storage," in *Proc. 45th Asilomar Conf. Signals, Syst., Comp.*, 2011, pp. 1850–1854.
[25] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "Xoring elephants: Novel erasure codes for big data," *Proc. VLDB Endow.*, vol. 6, no. 5, pp. 325–336, Mar. 2013.
[26] L. Pamies-Juarez, F. Blagojević, R. Mateescu, C. Gyuot, E. E. Gad, and Z. Bandić, "Opening the chrysalis: On the real repair performance of msr codes," in *14th USENIX Conference on File and Storage Technologies (FAST 16)*. USENIX Association, Feb. 2016, pp. 81–94.
[27] K. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A "hitchhiker's" guide to fast and efficient data reconstruction in erasure-coded data centers," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, ser. SIGCOMM '14. ACM, 2014, pp. 331–342.
[28] K. Rashmi, P. Nakkiran, J. Wang, N. B. Shah, and K. Ramchandran, "Having your cake and eating it too: Jointly optimal erasure codes for i/o, storage, and network-bandwidth," in *13th USENIX Conference on File and Storage Technologies (FAST 15)*. USENIX Association, Feb. 2015, pp. 81–94.
[29] Y. Hu, Y. Xu, X. Wang, C. Zhan, and P. Li, "Cooperative recovery of distributed storage systems from multiple losses with network coding," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 2, pp. 268–276, February 2010.
[30] X. Wang, Y. Xu, Y. Hu, and K. Ou, "Mfr: Multi-loss flexible recovery in distributed storage systems," in *IEEE International Conference on Communications (ICC)*, May 2010, pp. 1–5.
[31] A. M. Kermarrec, N. L. Scouarnec, and G. Straub, "Repairing multiple failures with coordinated and adaptive regenerating codes," in *International Symposium on Network Coding*, July 2011, pp. 1–6.
[32] J. Li and B. Li, "Cooperative repair with minimum-storage regenerating codes for distributed storage," in *IEEE Conference on Computer Communications (INFOCOM)*, April 2014, pp. 316–324.
[33] M. Ye and A. Barg, "Explicit constructions of optimal-access MDS codes with nearly optimal sub-packetization," *CoRR*, vol. abs/1605.08630, 2016.
[34] B. Sasidharan, G. K. Agarwal, and P. V. Kumar, "A high-rate msr code with polynomial sub-packetization level," in *Proc. IEEE Int. Symp. Inf. Theory*, 2015, pp. 2051–2055.
[35] M. Ye and A. Barg, "Explicit constructions of high-rate MDS array codes with optimal repair bandwidth," *CoRR*, vol. abs/1604.00454, 2016.

Balanced Locally Repairable Codes
Katina Kralevska, Danilo Gligoroski, and Harald Øverby
International Symposium on Turbo Codes and Iterative Information Processing, 2016

Paper 6

Balanced Locally Repairable Codes

Katina Kralevska, Danilo Gligoroski, and Harald Øverby
Department of Telematics; Faculty of Information Technology, Mathematics and Electrical Engineering; NTNU, Norwegian University of Science and Technology
Email: {katinak, danilog, haraldov}@item.ntnu.no

Abstract

We introduce a family of balanced locally repairable codes (BLRCs) $[n, k, d]$ for arbitrary values of n, k and d. Similar to other locally repairable codes (LRCs), the presented codes are suitable for applications that require a low repair locality. The novelty that we introduce in our construction is that we relax the strict requirement the repair locality to be a fixed small number l, and we allow the repair locality to be either l or $l + 1$. This gives us the flexibility to construct BLRCs for arbitrary values of n and k which partially solves the open problem of finding a general construction of LRCs. Additionally, the relaxed locality criteria gives us an opportunity to search for BLRCs that have a low repair locality even when double failures occur. We use metrics such as a storage overhead, an average repair bandwidth, a Mean Time To Data Loss (MTTDL) and an update complexity to compare the performance of BLRCs with existing LRCs.

Keywords: Locally Repairable codes, Balanced, Storage overhead, Update complexity, Repair bandwidth, MTTDL

I. Introduction

A conventional approach for achieving reliability in big data distributed storage systems is replication. In particular, the reliability of 3-replication is an accepted industry standard for management of hardware failures and recovery. That is an apparent situation in systems such as Hadoop HDFS [1], OpenStack SWIFT [2] or Microsoft Azure [3]. However, the accelerated and relentless data growth has made erasure coding a valuable alternative to 3-replication since erasure coding provides the same reliability as 3-replication, but with significant less storage overhead. Recently, there have been several proposals and experimental beta implementations of different types of erasure codes for huge distributed storage systems [4], [5].

Besides the reliability and the storage overhead, another important feature in distributed storage systems is the efficiency of the repair of a failed node. The efficiency is measured with two metrics: the repair bandwidth and the repair locality. The repair bandwidth is the amount of transferred data during a node repair, while the repair locality is the number of nodes contacted during the node repair process. Two types of erasure codes that address the repair efficiency have emerged: Regenerating codes [6] and Locally Repairable Codes (LRCs) [7]–[9].

Regenerating codes [6] aim to minimize the repair bandwidth, while LRCs seek to minimize the repair locality. The main idea behind regenerating codes is using a sub-packetization. Each block is divided into sub-packets and a recovery is performed by transferring sub-packets from all $n - 1$ non-failed nodes that results in high I/O. A proposal for reducing the I/O is given in [10]. On the other hand, LRCs address the issue of accessing less nodes, but the amount of transferred data is bigger compared to regenerating codes. However, communicating less nodes is beneficial for storage applications that require low I/O.

An $[n, k, d]_q$ MDS code C has to transfer k symbols to recover one lost symbol. LRCs were independently introduced in [7]–[9]. The code C has a locality l if the $i-$th code symbol c_i, $1 \leq i \leq n$, can be recovered by accessing l symbols where $l < k$. It was proved in [7] that the minimum distance of an $[n, k, d]_q$ code with a locality l is

$$d \leq n - k + 2 - \left\lceil \frac{k}{l} \right\rceil. \tag{1}$$

Huang et al. showed the existence of pyramid codes that achieve this distance when the field size is big enough [11].

Two practical LRCs have been implemented in Windows Azure Storage [12] and HDFS-Xorbas by Facebook [13]. Both implementations reduce the repair bandwidth and the I/O for reconstructing a single data block by introducing a fixed number of l local and r global parity blocks. Any single data block can be recovered from $\frac{k}{l}$ blocks within its local group. However, reconstruction of any global parity block (in Windows Azure) or double blocks failures is performed in the same way as with Reed-Solomon (RS) codes, i.e., k blocks need to be transferred.

Since node failures in storage systems are often correlated [14], there is a need for other erasure codes than LRCs for recovery from multiple failures. For instance, Shingled erasure codes (SHEC) have a low average repair bandwidth when multiple failures occur, but they are not so efficient in terms of the storage overhead and the reliability [15].

The locality also has an impact on the update complexity [16]. This is particularly important for hot data, i.e., frequently accessed data. For instance, an $[16, 10]$ LRC where the locality is 5, writing a data block takes 6 write operations (1 write to itself, 1 write to the local parity and 4 writes to the global parities).

Thus, having a general construction of LRCs that are simultaneously optimal in terms of storage overhead, reliability, locality and update complexity for a single failure and double failures is an important problem that is addressed in this work.

A. Our Contribution

We define a new family of balanced locally repairable codes (BLRCs). One of their main characteristics is that every systematic block has an equal (balanced) influence to the parity blocks. That is to say, each systematic block affects exactly w parity blocks. Additionally, we pay attention on the repair locality. In our construction we use a similar (but yet different) approach to the approaches introduced by Luby et al. for the construction of irregular LDPC codes [17] and Garcia-Frias and Zhong for the construction of regular and irregular LDGM codes [18]. Namely, instead of the strict requirement the repair locality to be a fixed small number l, it may be either l or $l + 1$. This partially solves the open problem given by Tamo et al. about a general construction of LRCs [19], because we construct LRCs for arbitrary values of n and k, but the locality is not strictly equal to l. Moreover, the relaxed locality criteria gives us an opportunity to search for BLRCs that have a low repair locality even when double failures occur. We use four metrics to examine the performance of BLRCs:

- Storage overhead (a ratio of the parity to the data blocks $\frac{r}{k}$);
- Average repair bandwidth (a ratio of the repair bandwidth to repair both data and parity blocks to the total stored data (sum of the data and the parity blocks));
- MTTDL (Mean Time To Data Loss - an estimate of the expected time that it would take a given storage system to exhibit enough failures such that at least one block of data cannot be retrieved or reconstructed);
- Update complexity (a maximum number of elements that must be updated when any single element is changed).

In summary, several goals are achieved simultaneously with this work: 1) low storage overhead; 2) low average repair bandwidth for single and double failures; 3) high reliability; and 4) improved update performance.

The paper is organized as follows. In Section II, we introduce the terminology and the definition of balanced locally repairable codes. In Section III, we give code examples and examine their performance by using the predefined metrics. We also compare the properties of our codes to 3-replication, RS and other LRCs. A reliability analysis is presented in Section IV. Conclusions are summarized in Section V.

II. Definition of Balanced Locally Repairable codes

We use the following notations throughout the rest of the paper. A file of size M is divided into k equally sized blocks and encoded in $GF(q)$ with an $[n, k, d]_q$ code into n coded blocks. An $[n, k, d]_q$ code is called

maximum distance separable (MDS) if $d = n - k + 1$. An $[n, k, d]_q$ MDS code reconstructs a failed block from any k out of the n blocks. We denote the number of parity blocks with $r = n - k$.

Definition 1: Let C be an $[n, k, d]_q$ code over $GF(q)$ with a generator matrix G:

$$G = \left[I_k | P \right], \tag{2}$$

where I_k is an identity matrix of order k and the $k \times (n-k)$ matrix P specifies how the parity is defined for the given $[n, k, d]_q$ linear code. We call C a *Balanced Locally Repairable Code (BLRC)*, if the Hamming weight of every row in the matrix P is w where $w < k$, the Hamming weight of every column is l or $l+1$ and for every submatrix P' of P consisting of v rows, $1 \leq v \leq w$, from P it holds that $Rank(P') = v$. The field size should be big enough so that the condition for the rank in Definition 1 is fullfiled.

Example 1: Let us consider the following $[13, 8, 3]$ code with a generator matrix:

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{1,11} & c_{1,12} & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & c_{2,9} & 0 & 0 & c_{2,12} & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & c_{3,9} & 0 & c_{3,11} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & c_{4,11} & 0 & c_{4,13} \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & c_{5,12} & c_{5,13} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & c_{6,10} & 0 & 0 & c_{6,13} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & c_{7,10} & c_{7,11} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & c_{8,9} & c_{8,10} & 0 & 0 & 0 \end{bmatrix},$$

where $c_{i,j}$ are some nonzero elements from $GF(q)$. Note that the Hamming weight of every row in P is $w = 2 < k$, while the Hamming weight of every column in P is either $l = 3$ or $l + 1 = 4$. Finally, since any two rows in P are linearly independent, the rank condition from Definition 1 is fulfilled. Thus, the code is a balanced locally repairable code.

From the erasure recovery point of view, we use the minimum distance of the code as a metric for its fault tolerance. We have the following Lemma:

Lemma 1: If $[n, k, d]_q$ is a balanced locally repairable code defined in a finite field $GF(q)$, then

$$d = w + 1. \tag{3}$$

Proof: The minimum distance d of a code C is equal to the number of failed blocks (erasures) after which the data cannot be recovered. Note that if one systematic block and w parity blocks that are linear combinations of the specific systematic block fail, then the systematic block is non-recoverable. This is true due to the fact that all $w + 1$ parts that have (non-encoded or encoded) information about the systematic block have been lost. Thus, it follows that $d \leq w + 1$. Let us assume that w_s systematic and w_p parity blocks are lost where $w = w_s + w_p$. If we consider that the lost w_p parity blocks are linear combinations from the w_s systematic blocks that have been also lost, then the systematic blocks can be recovered only if for every submatrix P' consisting of w_s rows of P it holds that $Rank(P') = w_s$. After recovering the systematic blocks, the lost parity blocks can be recovered. Let us consider that $w_s = w$ systematic blocks have been lost. The lost systematic blocks can be recovered by selecting the corresponding rows that contain the specific w systematic blocks in the matrix P and producing a matrix P'. Since $Rank(P') = w$, the lost systematic blocks can be recovered. On the other hand, if $w_p = w$ parity blocks have been lost, then each of the parity blocks can be recovered from l or $l + 1$ systematic blocks. In any case $d > w$. Consequently, it follows that $d = w + 1$. ■

The locality of a systematic code C is defined as the number of data blocks that each parity block is a function of.

Lemma 2: Let $[n, k, d]_q$ is a balanced locally repairable code defined in a finite field $GF(q)$. Then for its locality l, it holds:

$$l = \left\lfloor \frac{(d-1) \times k}{n-k} \right\rfloor. \tag{4}$$

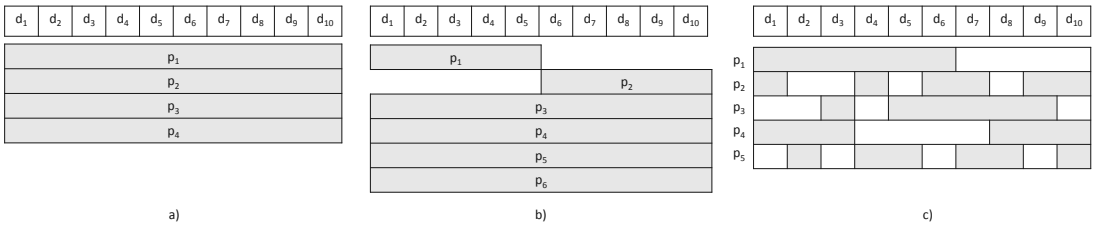


Figure 1: (a) An [14, 10] RS code where $l = 10$; (b) An [16, 10] LRC where $l = 5$ for the local parities p_1 and p_2 and $l = 10$ for the global parities p_3, p_4, p_5 and p_6; (c) An [15, 10] BLRC where $l = 6$ and $w = 3$

Proof: The parity part P of the generator matrix G is an $k \times (n-k)$ matrix. Since every row has w nonzero elements, with k such rows, the total number of nonzero elements in P is $w \times k$. It follows that the average number of nonzero elements in every column of P is $l = \left\lfloor \frac{(d-1)\times k}{n-k} \right\rfloor$. ■

The number of transferred blocks during a repair process and the update complexity for BLRCs are captured in the following propositions:

Proposition 1: When recovering one lost block (a systematic or parity block) in an $[n, k, d]_q$ balanced locally repairable code, the number of transferred blocks is l or $l + 1$.

The proof for Proposition 1 in connection with Lemma 2 includes a detailed algorithm how to construct BLRCs. We do not include it in this short paper due to space limitations, but we will include it in an extended version.

Proposition 2: The number of writes per update of an $[n, k, d]_q$ balanced locally repairable code is $w+1$.

Since the node failures in storage systems are often correlated [14], we next give an algorithm for finding BLRCs that have a low repair locality even when two blocks have failed. Algorithm 1 uses a stochastic hill-climbing search in a similar manner as in [20], [21].

Algorithm 1 A general Stochastic Hill-Climbing search for finding a locally repairable code for given n, k and d

Input: n, k and d;

Output: A Balanced Locally Repairable Code.

1: Find a random $[n, k, d]$ linear code as in Definition 1 where $w = d-1$ is the Hamming weight of every row of the matrix P;

2: Repeatedly improve the solution by searching for codes with low average locality when two blocks failures have to be recovered, until no more improvements are necessary/possible.

The construction of our codes has some similarities with the construction of several classes of LDPC codes reported in the literature. In particular, several families of LDPC codes that are based on Finite Geometries are defined in [19]. In that work, the restrictions that are inferred by the properties of Finite Geometries restrict the possible choices of different n and k. Variable irregular LDPC codes are constructed by puncturing the codes or by splitting the columns and rows of the parity-check matrix H.

We have been inspired by two other works that are also from the area of LDPC codes. Namely, Luby et al. in 2001 introduced the principle of allowing irregularities for variable nodes in a LDPC construction [17]. They allowed those irregularities to have degree 2, 3, 4 or even 20, while in our construction the degree of locality is either l or $l + 1$. On the other hand, in 2003 Garcia-Frias and Zhong proposed regular and irregular LDGM codes in [18]. For the regular LDGM codes, the parity matrix P has always a fixed row weight X and fixed column weight Y which is equivalent to the LRC case where the row weight is fixed

at w and the column weight is fixed at l. For the irregular LDGM codes the parity matrix P has an average row weight X and an average column weight Y, while in our construction the row weight is fixed to w but the column weight can be either l or $l+1$.

III. Examples of Code Constructions

In this Section we present several parity parts P (not to be confused with a parity-check matrix H) of BLRCs for different code parameters.

The parity part P_1 of an $[15, 10]$ code for $l = 6$ and $w = 3$ is:

$$P_1 = \begin{bmatrix} c_{1,11} & c_{1,12} & 0 & c_{1,14} & 0 \\ c_{2,11} & 0 & 0 & c_{2,14} & c_{2,15} \\ c_{3,11} & 0 & c_{3,13} & c_{3,14} & 0 \\ c_{4,11} & c_{4,12} & 0 & 0 & c_{4,15} \\ c_{5,11} & 0 & c_{5,13} & 0 & c_{5,15} \\ c_{6,11} & c_{6,12} & c_{6,13} & 0 & 0 \\ 0 & c_{7,12} & c_{7,13} & 0 & c_{7,15} \\ 0 & 0 & c_{8,13} & c_{8,14} & c_{8,15} \\ 0 & c_{9,12} & c_{9,13} & c_{9,14} & 0 \\ 0 & c_{10,12} & 0 & c_{10,14} & c_{10,15} \end{bmatrix},$$

where the coefficients $c_{i,j}$, $i \leq 10$ and $11 \leq j \leq 15$, are elements from a finite field $GF(q)$. Since we do not show the I_k matrix, the index j for the non-zero coefficients is in the range between $k+1$ and n. Note that the number of non-zero elements in P_1 per row is $w = 3$ and per column is $l = 6$. A transposed P_1 is graphically represented in Figure 1c. The non-zero elements are represented with the shaded blocks in Figure 1c. The average repair bandwidth for a single failure is 6 and for double failures is 9. For comparative purposes we graphically represent the parity parts of an [14, 10] RS and an [16, 10] LRC in Figure 1a and 1b, respectively. As we can see the RS code has the biggest locality $l = 10$. Consequently, a transfer of 10 blocks is required to repair any systematic or parity block when the RS code is used. The [16, 10] LRC has locality equal to 5 for the local parity blocks and 10 for the global parity blocks. Therefore, it requires a transfer of 5 blocks to repair a single failure of the systematic and the local parity blocks, while it takes 10 blocks to repair the global parities. Hence, the average repair bandwidth for a single failure with the [16, 10] Azure LRC is $(5 \times 12 + 10 \times 4)/16 = 6.25$. However, the [16, 10] Xorbas LRC reduces the number of transferred blocks for a repair of any single global parity block to 5 by introducing an implied parity block. Thus, it has a lower average repair bandwidth compared to the Azure LRC implementation. A comparison of the performance metrics for the $[15, 10]$ code with parity part P_1 with 3-replication, the [14, 10] RS and the [16, 10] Xorbas LRC is presented in Table I, while additionally the Azure LRC is added in Figure 2. The way how we calculate the MTTDL is described in Section IV.

The next example of an [16, 10] code for $l = 5$ and $w = 3$ shows even a better performance when double failures occur. The average repair bandwidth for a single failure is 5, while it is 6.75556 for double failures. This code tolerates up to any 3 failures and recovers the data successfully with 99.45%, 96.02% and 79.66% from 4, 5 and 6 failures, respectively. Its parity part is given as

$$P_2 = \begin{bmatrix} c_{1,11} & 0 & 0 & 0 & c_{1,15} & c_{1,16} \\ 0 & c_{2,12} & c_{2,13} & 0 & 0 & c_{2,16} \\ 0 & c_{3,12} & 0 & 0 & c_{3,15} & c_{3,16} \\ 0 & 0 & c_{4,13} & c_{4,14} & c_{4,15} & 0 \\ c_{5,11} & 0 & 0 & c_{5,14} & 0 & c_{5,16} \\ 0 & c_{6,12} & c_{6,13} & c_{6,14} & 0 & 0 \\ c_{7,11} & c_{7,12} & 0 & c_{7,14} & 0 & 0 \\ c_{8,11} & 0 & 0 & c_{8,14} & c_{8,15} & 0 \\ 0 & 0 & c_{9,13} & 0 & c_{9,15} & c_{9,16} \\ c_{10,11} & c_{10,12} & c_{10,13} & 0 & 0 & 0 \end{bmatrix}.$$

We depict the average repair bandwidth for different number of block failures for the RS, Xorbas LRC, Azure LRC and BLRCs in Figure 2, under the condition of almost equal MTTDL. BLRCs achieve an

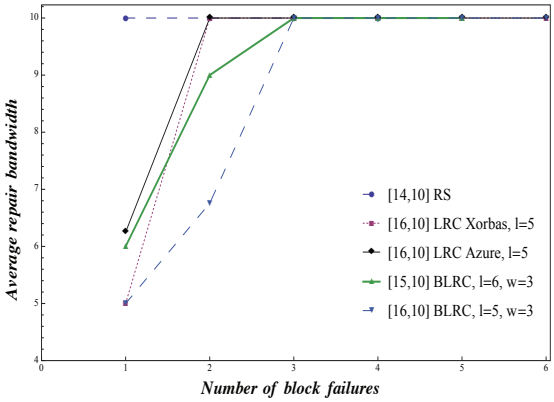


Figure 2: Average repair bandwidth for different number of block failures

average repair bandwidth that is less than or equal to the average repair bandwidth with the other codes in case of a single block failure, while it is always less than the average repair bandwidth achieved with the other codes in case of double block failures. Note that the storage overhead is less with the BLRC compared to the Xorbas LRC when they achieve the same average repair bandwidth for a single failure.

The next example shows how the repair bandwidth can be reduced even more, but then the fault tolerance is worse. This has a direct impact on the reliability, i.e., MTTDL. The parity part of an $[16, 10]$ code for $l = 3$ or $l = 4$ and $w = 2$ is

$$P_3 = \begin{bmatrix} c_{1,11} & 0 & 0 & 0 & 0 & c_{1,16} \\ 0 & 0 & 0 & c_{2,14} & c_{2,15} & 0 \\ 0 & 0 & 0 & c_{3,14} & 0 & c_{3,16} \\ 0 & 0 & 0 & 0 & c_{4,15} & c_{4,16} \\ c_{5,11} & 0 & 0 & c_{5,14} & 0 & 0 \\ 0 & c_{6,12} & 0 & 0 & c_{6,15} & 0 \\ c_{7,11} & c_{7,12} & 0 & 0 & 0 & 0 \\ c_{8,11} & 0 & c_{8,13} & 0 & 0 & 0 \\ 0 & 0 & c_{9,13} & 0 & c_{9,15} & 0 \\ 0 & c_{10,12} & c_{10,13} & 0 & 0 & 0 \end{bmatrix}.$$

When applying this code the average repair bandwidth for a single block failure is reduced to 3.33, while for double block failures to 5.22. On the other hand, the fault tolerance is worse compared to the [16, 10] code for $l = 5$ and $w = 3$. Thus, the MTTDL is reduced from 5.7378×10^{14} days to 7.2338×10^{8} days for an [16, 10] code when $l = 5$, $w = 3$ and $l = 3.33$, $w = 2$, respectively.

An overview of the performance metrics for the codes presented in this Section is given in Table I.

IV. Reliability Analysis

We perform a reliability analysis by calculating the MTTDL with a Markov model. The authors in [13] report values from the Facebook cluster and show that the $[16, 10]$ Xorbas LRC provides significantly longer MTTDL compared to the $[14, 10]$ RS and 3-replication.

In our analysis, we use the same parameters as in [13] in order to compare the results. The total size of the cluster data is $C = 30PB$ and this data is stored in $N = 3000$ nodes. The mean time to failure of a disk node is 4 years (=1/λ) and the block size is $B = 256MB$. The node failures are independent. The bandwidth for cross-rack communication for repairs is limited to $\gamma = 1Gbps$. Under an $[15, 10]$ code, each stripe consists of 15 blocks where each block is placed in different racks to provide a higher fault tolerance. Thus the total number of stripes in the system is $C/(nB)$ where $n = 15$. The MTTDL of a stripe

TABLE I: Comparison summary of performance metrics for 3-replication, RS, Xorbas LRC and BLRCs

Scheme	Storage overhead	Avr. repair bandwidth (single failure)	Avr. repair bandwidth (double failure)	MTTDL (days)	Update complexity
3-replication	2x	1x	1x	2.3079×10^{10}	3
[14, 10] RS code	0.4x	10x	10x	3.3118×10^{13}	5
[16, 10] Xorbas LRC, $l = 5$	0.6x	5x	10x	1.2180×10^{15}	6
[15, 10] BLRC, $l = 6, w = 3$	0.5x	6x	9x	3.3647×10^{14}	4
[16, 10] BLRC, $l = 5, w = 3$	0.6x	5x	6.76x	5.7378×10^{14}	4
[16, 10] BLRC, $l = 3$ or $4, w = 2$	0.6x	3.33x	5.22x	7.2338×10^{8}	3

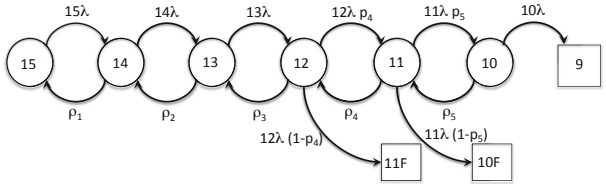


Figure 3: Markov model for an [15, 10] code where circles represent the states when the data can be recovered and squares represent the states when the data is unrecoverable

is calculated by using the Markov model shown in Figure 3. Each state in the Markov model represents the number of available (non-failed) blocks (data and parity blocks). The circles denote the states when the system is up and the squares denote the states when there is a data loss in the system, i.e., the system is down.

Let λ denotes the failure rate of a single block. The blocks are distributed in different nodes and the failure rate per node is λ. When the state is i, i.e., there are i available blocks in a stripe, the failure rate is $i\lambda$. Consequently, the transition rate from State 15 to State 14 is 15λ. Since BLRCs are not MDS codes, there are two possible transitions from State 12. One of the transitions is to State 11 where there are 4 decodable failures and the other one is to State 11F which represents a state with 4 non-decodable failures. The percentage of 4 decodable failures is $p_4 = 99.2674\%$. Therefore the transition rate to State 11 is $12\lambda p_4$ and to State 11F is $12\lambda(1 - p_4)$. The same situation happens when transitioning from State 11 to State 10 and State 10F where $p_5 = 89.677\%$. When 6 blocks are lost, i.e., only 9 blocks are available in State 9, the lost blocks from the stripe cannot be recovered. That is why State 9 is shown as a down state. The lost blocks from the stripe are also non-recoverable in the States 11F and 10F.

In the reverse direction ρ_i denotes the repair rate. The rate at which a block is repaired depends on the number of downloaded blocks (the locality), the block size and the bandwidth dedicated for repairs. For instance, a repair of any single data or parity block requires downloading 6 blocks, i.e., $\rho_1 = \frac{\gamma}{6B}$. Any two lost blocks are repaired by downloading 9 blocks, while a repair of more than 2 lost blocks requires a transfer of 10 blocks. Thus, $\rho_2 = \frac{\gamma}{9B}$ and $\rho_3 = \rho_4 = \rho_5 = \frac{\gamma}{10B}$. The MTTDL of the system is calculated as:

$$MTTDL = \frac{MTTDL_{stripe}}{C/(nB)}. \tag{5}$$

The MTTDL values for 3-replication, the [14, 10] RS, the [16, 10] Xorbas LRC and few BLRCs are presented in Table I. We observe that the fast repair and the high fault tolerance lead to a high reliability with the [15, 10] and [16, 10] BLRCs.

V. Conclusions

We defined a new family of balanced locally repairable codes (BLRCs). A novel property of the codes that we presented is that there is no strict requirement that the repair locality is a fixed small number l, and

it may be either l or $l+1$. Advantageously, this provides the flexibility to construct BLRCs for arbitrary values of n and k which allows a general construction of LRCs. The properties of the presented codes are: low storage overhead, low average repair bandwidth for a single failure and double failures, high reliability and low update complexity.

REFERENCES

[1] D. Borthakur, "The hadoop distributed file system: Architecture and design," Hadoop Project Website, 2007.

[2] J. Arnold, "Openstack swift: Using, administering, and developing for swift object storage," 2014.

[3] B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y. Xu, S. Srivastav, J. Wu, H. Simitci et al., "Windows azure storage: a highly available cloud storage service with strong consistency," in 23rd ACM Symposium on Operating Systems Principles, 2011, pp. 143–157.

[4] B. Fan, W. Tantisiriroj, L. Xiao, and G. Gibson, "Diskreduce: Raid for data-intensive scalable computing," in Proceedings of the 4th Annual Workshop on Petascale Data Storage, 2009, pp. 6–10.

[5] J. S. Plank, S. Simmerman, and C. D. Schuman, "Jerasure: A library in c/c++ facilitating erasure coding for storage applications-version 1.2," Tech. Rep., 2008.

[6] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," IEEE Transactions on Information Theory, vol. 56, no. 9, Sept 2010, pp. 4539–4551.

[7] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," IEEE Transactions on Information Theory, vol. 58, no. 11, 2012, pp. 6925–6934.

[8] F. E. Oggier and A. Datta, "Self-repairing homomorphic codes for distributed storage systems," in INFOCOM, 2011, pp. 1215–1223.

[9] D. Papailiopoulos, J. Luo, A. Dimakis, C. Huang, and J. Li, "Simple regenerating codes: Network coding for cloud storage," in IEEE INFOCOM, 2012, pp. 2801–2805.

[10] O. Khan, R. C. Burns, J. S. Plank, W. Pierce, and C. Huang, "Rethinking erasure codes for cloud file systems: minimizing I/O for recovery and degraded reads," in Proceedings of the 10th USENIX conference on File and Storage Technologies, 2012.

[11] C. Huang, M. Chen, and J. Li, "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," in IEEE International Symposium on Network Computing and Applications, July 2007, pp. 79–86.

[12] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in windows azure storage," in USENIX Annual Technical Conference, 2012, pp. 15–26.

[13] M. Sathiamoorthy, M. Asteris, D. S. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "XORing elephants: Novel erasure codes for big data," vol. 6, no. 5, 2013, pp. 325–336.

[14] D. F., F. Labelle, F. I. Popovici, M. Stokely, V. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in globally distributed storage systems," in 9th USENIX Symposium on Operating Systems Design and Implementation, 2010, pp. 61–74.

[15] T. Miyamae, T. Nakao, and K. Shiozawa, "Erasure code with shingled local parity groups for efficient recovery from multiple disk failures," in 10th Workshop on Hot Topics in System Dependability. USENIX Association, 2014.

[16] N. Anthapadmanabhan, E. Soljanin, and S. Vishwanath, "Update-efficient codes for erasure correction," in 48th Annual Allerton Conference on Communication, Control, and Computing, Sept 2010, pp. 376–382.

[17] M. G. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman et al., "Improved low-density parity-check codes using irregular graphs," IEEE Transactions on Information Theory, vol. 47, no. 2, 2001, pp. 585–598.

[18] J. Garcia-Frias and W. Zhong, "Approaching shannon performance by iterative decoding of linear codes with low-density generator matrix," IEEE Communications Letters, vol. 7, no. 6, 2003, pp. 266–268.

[19] I. Tamo, D. S. Papailiopoulos, and A. G. Dimakis, "Optimal locally repairable codes and connections to matroid theory," in IEEE International Symposium on Information Theory, 2013, pp. 1814–1818.

[20] Y. Wang, J. S. Yedidia, and S. C. Draper, "Construction of high-girth qc-ldpc codes," in 5th International Symposium on Turbo Codes and Related Topics, Sept 2008, pp. 180–185.

[21] D. Gligoroski and K. Kralevska, "Families of optimal binary non-mds erasure codes," in IEEE International Symposium on Information Theory, June 2014, pp. 3150–3154.

Coded Packet Transport for Optical Packet/Burst Switched Networks

Katina Kralevska, Harald Øverby, and Danilo Gligoroski

IEEE Proceedings on Global Communications Conference (GLOBECOM),
pp. 1-6, 2015

Paper 7

Coded Packet Transport for Optical Packet/Burst Switched Networks

Katina Kralevska, Harald Øverby, and Danilo Gligoroski
Department of Telematics
Norwegian University of Science and Technology, Trondheim, Norway,
Email: katinak@item.ntnu.no, haraldov@item.ntnu.no, danilog@item.ntnu.no

Abstract

This paper presents the Coded Packet Transport (CPT) scheme, a novel transport mechanism for Optical Packet/Burst Switched (OPS/OBS) networks. The CPT scheme exploits the combined benefits of source coding by erasure codes and path diversity to provide efficient means for recovering from packet loss due to contentions and path failures, and to provide non-cryptographic secrecy. In the CPT scheme, erasure coding is employed at the OPS/OBS ingress node to form coded packets, which are transmitted on disjoint paths from the ingress node to an egress node in the network. The CPT scheme allows for a unified view of Quality of Service (QoS) in OPS/OBS networks by linking the interactions between survivability, performance and secrecy. We provide analytical models that illustrate how QoS aspects of CPT are affected by the number of disjoint paths, packet overhead and processing delay.

Keywords: optical packet/burst switching, source coding, survivability, secrecy, performance

I. Introduction

Optical Packet/Burst Switching (OPS/OBS) is a promising architecture for the future core network, enabling all-optical packet transport combined with statistical multiplexing for increased link utilization [6]. By avoiding electronic processing of packets, OPS/OBS achieves significant energy savings compared to existing opaque packet switched architectures [24]. The increasing number of mission-critical services such as e-banking, e-voting and emergency services put a high demand on the Quality of Service (QoS) of the future Internet, including OPS/OBS networks. Specifically, the OPS/OBS network has to provide low packet loss rate (performance) [18], [19], [26], [27], protection against node and link failures (survivability) [5], [20], as well as being able to withstand targeted eavesdropping attacks from individuals and organizations (secrecy) [11].

Existing approaches to satisfy these strict QoS demands in OPS/OBS rely on the provision of several independent QoS schemes, e.g., wavelength conversion to reduce packet loss from contentions and 1+1 path protection to provide survivability. However, since these schemes are deployed in the same physical and logical infrastructure, they will interact and provide mutual benefits. Examples of this include how the extra redundancy introduced for providing 1+1 path protection may be used to combat packet loss in OPS networks in failure-free operations, as studied in [20]. In particular, security threats in all-optical networks have recently received research attention [3], [12]. One crucial security threat is eavesdropping of data in the network, which has traditionally been countered using encryption. However, the high capacities of OPS/OBS networks greater than 100 Gb/s make data encryption in OPS/OBS not feasible as the current computational resources do not match the required encryption processing demands. Hence, there is a need for a low complexity scheme that provides a certain level of secure data transport without encrypting the data. Our goal is to show how erasure coding and path diversity can be used to mutually provide loss recovery from contentions, survivability and a secrecy of data.

The major contribution of this paper is the novel Coded Packet Transport (CPT) scheme for OPS/OBS networks. This scheme is able to recover lost data due to contentions and node/link failures, while at the

same time providing secrecy. We use the term secrecy as defined in [11], where the goal is protection from a passive adversary that is not able to reconstruct the whole packet/burst set by eavesdropping on a single path. The CPT scheme is based on Forward Error Correction (FEC) codes used as erasure codes and provides non-cryptographic secrecy. The CPT scheme is applicable to both OPS and OBS networks, and for the remainder of the paper we use the term packet to also refer to a burst in OBS networks, without the loss of generality. At an OPS/OBS ingress node, a set of data packets is encoded into a set of coded packets by utilizing non-systematic erasure codes [9], [17], [25]. These coded packets are transmitted to an egress node in the OPS/OBS network on multiple disjoint paths. At an OPS/OBS egress node, reconstruction of packets lost due to contentions and node/link failures is enabled by the added redundancy. Sending different subsets of packets over disjoint paths between the ingress and the egress node also enables an end-to-end secrecy property against a passive adversary. To the best of our knowledge, this work constitutes a first step for providing a unified view on QoS in OPS/OBS networks, focusing on the interactions between survivability, performance and secrecy.

The rest of this paper is organized as follows: Section II discusses related works. In Section III we present the CPT scheme. Section IV presents the analytical model. The parameter settings based on the analytical model are presented in Section V. Finally, Section VI concludes the paper.

II. Related works

The authors of [17] and [25] show how FEC codes can be used to reduce packet loss in OPS networks. Here, redundant packets are added to a set of data packets at the OPS ingress node and transmitted along with the original data packets to an OPS egress node. Data packets dropped due to contentions may be reconstructed at the OPS egress node by using excess redundant packets, leading to a potential reduced Packet Loss Rate (PLR). The work in [20] and [16] extends these schemes to provide 1+1 path protection. One redundant packet is added to a packet set using the XOR operation. This packet set is transmitted to the OPS egress node over three or more disjoint paths. In particular, the authors of [16] evaluate this scheme from a cost perspective, comparing it to other approaches that provide 1+1 path protection, showing that significant cost savings can be achieved using erasure codes. Unlike the present paper, these schemes do not consider secrecy.

The authors of [14] show how the PLR can be significantly reduced by sending packets at appropriate rates on disjoint paths from multiple ingress nodes to an egress node by using FEC techniques. The work is further extended in [13] where a scalable, heuristic scheme for selecting a redundant path between an ingress node and an egress node is presented. Another way of reducing the packet loss due to contentions in OPS networks is by combining source and network coding. Instead of dropping the colliding packets at the intermediate node, they are XOR-ed together [2], [7].

The authors of [3] suggest an OBS framework that provides authentication of burst headers and confidentiality of data bursts based on encryption. However, due to the high bandwidth in OBS networks, the encryption mechanisms have to be with low computational complexity, suitable for high-speed implementation and the majority of the header content should not be encrypted since the processing of the headers has to be at ultra high speed [3]. Unlike their work, we provide a certain level of non-cryptographic secrecy in the data transport without using encryption, thus significantly reducing the computational complexity.

That secrecy is achieved by sending non-systematic coded packets on disjoint paths. This property has so far not been exploited in OPS/OBS networks. The authors of [15] show how to provide secrecy in storage systems even when an eavesdropper knows or can guess some of the missing symbols. This is achieved by using MDS matrices (matrices that have no singular square submatrices). On the other hand, schemes based on non-systematic codes increase the delay in the networks, as decoding of non-systematic coded packets can start after the number of received coded packets is at least equal to the number of original data packets. In addition, the encoding and decoding of the packets also add processing delay in the network.

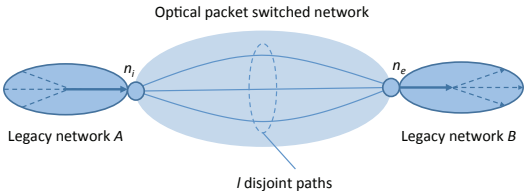


Fig. 1: An OPS network where data is transmitted from ingress node n_i to egress node n_e

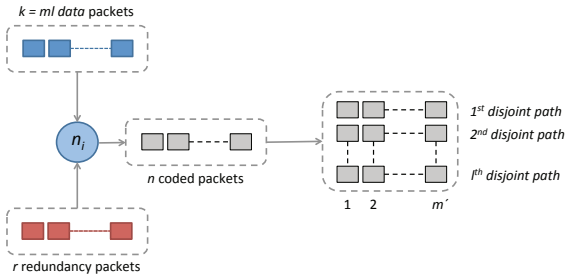


Fig. 2: Illustration of CPT, k data packets are encoded into n coded packets at n_i and transmitted over l disjoint paths

III. Coded Packet Transport (CPT)

We focus on an ingress and egress node pair (n_i, n_e) in an OPS/OBS network as depicted in Fig. 1. Packets arrive at n_i from a legacy network A, with destination n_e or a legacy network B. We assume that there exist l disjoint paths between n_i and n_e. We use the term disjoint paths to refer to node disjoint (respectively link disjoint) paths between two nodes in the network. The ingress node n_i encodes the k data packets into n coded packets. The disjoint paths are independent and disjunctive subsets of the coded packets are sent on them. Figure 2 depicts the encoding of k data packets with equal length into n coded packets. Since we analyze the effect of l on the QoS in OPS/OBS, both the original and coded packets are written as multiple of l, i.e., $k = ml$ and $n = m'l$. Here we introduce the metric packet overhead o that is a ratio of the number of redundant packets and data packets, i.e., $o = \frac{r}{k}$. Notations used in the paper are summarized in Table I. Packets in the network may be lost due to contention inside packet switches and due to node/link failures [20]. Note that we exclude the ingress and egress node from the set of nodes that can fail.

The CPT scheme is based on FEC codes that are used as erasure codes. Different coding schemes may be applied for the CPT, including Maximum Distance Separable (MDS) and non-MDS codes. Let us denote by $GF(2^q)$ the Galois field with 2^q elements. An $(n, k, d)_{2^q}$ code is an $(n, k)_{2^q}$ code of length n and rank k with minimum weight d among all nonzero codewords. An $(n, k, d)_{2^q}$ code is called MDS if $d = n - k + 1$. The Singleton defect of an $(n, k, d)_{2^q}$ code C defined as $s(C) = n - k + 1 - d$ measures how far away is C from being MDS. Reed-Solomon codes are the most commonly used MDS codes, i.e., their Singleton defect is zero [22]. When Reed-Solomon codes are used, at least k out of n packets must arrive successfully at the egress node in order to enable recovery of the data.

An (n, k) linear block code is defined by its $n \times k$ generator matrix $\mathbf{G}$. A code is systematic if the first k rows of its generator matrix $\mathbf{G}$ contain the identity matrix. That means that a systematic linear code does not transform the original k data packets, but generates only extra $n - k$ redundant packets.

If the generator matrix $\mathbf{G}$ is not systematic then the code is non-systematic, and all n generated packets linearly depend on all original k packets via $\mathbf{G}$. Systematic codes are less processing demanding than non-systematic, since they do not require processing of the original data. If the goal is to achieve a certain

TABLE I: Overview of parameters

n_i	Ingress node
n_e	Egress node
k	Number of original data packets
n	Number of coded packets
r	Number of redundant packets
L	Packet length
l	Number of disjoint paths
m	Number of data packets sent on a disjoint path
m'	Number of coded packets sent on a disjoint path
$GF(2^q)$	Galois field
p	Packet Loss Rate
p_{thres}	Packet Loss Rate threshold
o	Packet overhead

level of secrecy in the transmitted data, then non-systematic codes should be used. That is the main reason why we use non-systematic Reed-Solomon codes.

An (n, k) Reed-Solomon code is obtained by evaluating polynomials over $GF(2^q)$ at n different points $\alpha_1, \alpha_2, \ldots, \alpha_n$. The generator matrix for Reed-Solomon code is an $n \times k$ Vandermonde matrix

$$\mathbf{G} = \begin{bmatrix} 1 & \alpha_1 & {\alpha_1}^2 & \ldots & {\alpha_1}^{k-1} \\ 1 & \alpha_2 & {\alpha_2}^2 & \ldots & {\alpha_2}^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_{n-1} & {\alpha_{n-1}}^2 & \ldots & {\alpha_{n-1}}^{k-1} \\ 1 & \alpha_n & {\alpha_n}^2 & \ldots & {\alpha_n}^{k-1} \end{bmatrix}.$$

The encoding in CPT is performed in the following way.

We assume that the k original data packets with equal length consist of L symbols from $GF(2^q)$, i.e., $x_i = (s_{i,j})$ where $1 \leq i \leq k$ and $1 \leq j \leq L$. The k data packets are presented in a form of a matrix $\mathbf{X}$ where every row represents one data packet

$$\mathbf{X} = \begin{bmatrix} s_{1,1} & s_{1,2} & \ldots & s_{1,L} \\ s_{2,1} & s_{2,2} & \ldots & s_{2,L} \\ \vdots & \vdots & \ddots & \vdots \\ s_{k,1} & s_{k,2} & \ldots & s_{k,L} \end{bmatrix}.$$

We obtain the coded packets by simple matrix multiplication of $\mathbf{G}$ with $\mathbf{X}$, i.e.,

$$\mathbf{Y} = \mathbf{G} \times \mathbf{X}.$$

Similarly as in $\mathbf{X}$, every row of $\mathbf{Y}$ represents an encoded packet. For the sake of clarity we represent the matrix $\mathbf{Y}$ in l stripes of $m' = \lceil \frac{n}{l} \rceil$ rows, since we consider transmitting data on l disjoint paths

$$\mathbf{Y} = \left[\begin{array}{cccc} s_{1,1} & s_{1,2} & \cdots & s_{1,L} \\ \vdots & \vdots & \ddots & \vdots \\ s_{m',1} & s_{m',2} & \cdots & s_{m',L} \\ \hline s_{m'+1,1} & s_{m'+1,2} & \cdots & s_{m'+1,L} \\ \vdots & \vdots & \ddots & \vdots \\ s_{2m',1} & s_{2m',2} & \cdots & s_{2m',L} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline s_{(l-1)m'+1,1} & s_{(l-1)m'+1,2} & \cdots & s_{(l-1)m'+1,L} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n,1} & s_{n,2} & \cdots & s_{n,L} \end{array}\right].$$

The packets from the $i-$th stripe are sent on the $i-$th disjoint path, respectively. Once again we emphasize that sending linear combinations of the data packets instead of using a systematic code offers secrecy as opposed to systematic codes.

Next we discuss the required parameter settings for enabling survivability, secrecy and performance within a predefined PLR threshold.

IV. Analytical model

A. Topology constraints

The parameter l dictates the number of disjoint paths between a pair of nodes in the network. We choose to vary l between 2 and 6, which is grounded in the constraints provided by most empirical network topologies. Hence, $1 + N$ protection for a big N is impractical. In the following analysis we combine path diversity with source coding to meet the goals of the CPT.

B. Survivability constraints

In order to provide 1+1 path protection against single link failure, the number of received packets at n_e must be equal to or larger than the original number of data packets, k. Hence, as the number of lost data packets in the case of a failed path is m', we have that

$$n - m' \geq k, \tag{1}$$

resulting in the following constraint for o

$$o \geq \frac{1}{l-1}. \tag{2}$$

C. Secrecy constraints

The goal is to achieve secrecy so that a passive adversary is not able to reconstruct the whole packet set by eavesdropping on a single path. We use here the term secrecy as it is used in [11, Ch.7 pp. 185].

An eavesdropper needs to eavesdrop k packets in order to decode the whole packet set. By eavesdropping on a single path, m' packets are obtained. Hence, we ensure that by eavesdropping on a single path it is not possible to recover the data packets if the following

$$k \geq m' \tag{3}$$

is fullfiled. Resulting in the following constraint for o

$$o \leq l - 1. \tag{4}$$

One straightforward attack on the secrecy of CPT is by applying the strategy described by McEliece and Sarwate in [10]. The authors conclude that decoding algorithms for Reed-Solomon codes provide extensions and generalizations of the Shamir's secret sharing scheme [23].

We adapt the strategy discussed in the last paragraph of [10]. Namely, the attack is a combination of a partially known coded text and a brute force attack. An eavesdropper knows or guesses the format or even the content of some parts of the coded information and performs an exhaustive search to check all possible linear combinations in order to filter out the wrong guesses and to find the correct ones. We give a short example where we show that constraint 4 is not enough for providing secrecy when an eavesdropper is able to guess the missing packets, i.e., run a brute force attack.

Example 1: Based on constraints 2 and 4, the network is in the operational range of CPT if $l = 3$ and $o = 1$. We define a range as operational when the secrecy and the survivability constraints are not violated. Let us take the following Reed-Solomon code (12, 6) over $GF(2^8)$. If we want to achieve survivability and secrecy, 4 packets are sent on each disjoint path. However, if an eavesdropper gets the coded packets from a single path, he/she gets 4 coded packets. The eavesdropper requires 2 more coded packets to decode the whole data set. Since the coding is performed over $GF(2^8)$, it applies an exhaustive search of $2^{2\times 8} = 2^{16}$ tries for every 2 missing bytes from the 2 missing packets. Knowing the format of the submitted information (which is a reasonable and plausible assumption) the eavesdropper filters out all solutions that do not fit within its filtering strategy and keeps the solutions that give decoded information that has the expected format.

We combine the attack technique discussed above with the modern recommended levels of security as they are given in [1], [4], [8]. The minimum level of security for 2015 is between 80 and 112 bits (Lenstra/Verhul, ECRYPT II, NIST), but we put a level of 128 bits as a long term security level. So we derive an additional constraint for secrecy in CPT considering these recommendations, i.e.,

$$(k - m')q \geq 128 \tag{5}$$

resulting in the following constraint for o

$$o \leq l - 1 - \frac{128}{qm}. \tag{6}$$

Example 2: In this example we still assume that l=3 and o=1. The coding is performed with Reed-Solomon over $GF(2^8)$, but n and k are chosen following constraint 6. Let us take the Reed-Solomon code (96, 48) over $GF(2^8)$. On each disjoint path 32 packets are sent. If an eavesdropper gets the coded packets on a single path, still it misses 16 packets to decode the original data. Running an exhaustive search of $2^{16\times 8} = 2^{128}$ tries for every 16 missing bytes from the 16 missing packets is infeasible.

D. Performance constraints

When the PLR is p and the loss of the redundant packets is accounted, then the average number of redundant packets is

$$r = \frac{kp}{1-p}. \tag{7}$$

However, adding r redundant packets does not guarantee that the PLR is kept bellow a predefined threshold p_{thres}.

Unsuccessful decoding occurs when more than r packets are lost. The probability of unsuccessful decoding p_{fail} when RS codes are used is

$$p_{fail} = \sum_{i=n-k+1}^{n} \binom{n}{i} p^i (1-p)^{n-i}. \tag{8}$$

If one path has failed, still data packets on the non-failed paths can be lost due to contentions. The required number of redundant packets is calculated for $n' = n - m'$ and $p_{fail} \leq p_{thres}$, i.e., from the following expression

$$\sum_{i=n'-k+1}^{n'} \binom{n'}{i} p^i (1-p)^{n'-i} \leq p_{thres}. \tag{9}$$

E. Processing constraints

Since in CPT we add redundancy of $n-k$ packets and we use non-systematic RS codes, we introduce processing delay. The processing delay is expressed in number of clock cycles.

The processing delay should not be confused with latency, because it is a component of the latency. Since decoding is more computationally demanding than encoding, we calculate the processing delay at the egress node. The processing delay is defined as the number of clock cycles required for a packet set to enter, be processed and leave the egress node.

In [21] the processing delay in number of clock cycles per symbol for a RS systematic code (n, k) at the decoder is calculated as

$$d_{proc} = (n-k)^2 + 6(n-k) + 4. \tag{10}$$

If the processing delay for the packet set is greater than the size of the packet set, $d_{proc} > n$, then the node should buffer the packets arriving from the next packet set while processing the previous one.

The time to process the packet set for a systematic code is dependent on the number of redundant packets $n-k$. For a non-systematic code, the size of the whole packet set n should be counted, not just $n-k$. First we obtain the delay for processing a single packet and then we multiply it with the total number of packets in the packet set. Therefore, the processing delay in number of clock cycles per symbol for a RS non-systematic code (n, k) at the decoder is

$$d_{proc} = \lceil \frac{n}{n-k} \rceil ((n-k)^2 + 6(n-k) + 4). \tag{11}$$

V. PARAMETER SETTINGS FOR CPT

Figure 3 shows the operational range of the CPT, i.e., the relationship between the number of available disjoint paths between a node pair (l) and the packet overhead (o). Three different areas are identified:

1) The operational range for the CPT scheme;
2) The secrecy constraints are violated;
3) The survivability constraints are violated.

As a general insight, we observe that a certain amount of packet overhead needs to be provided in order to ensure survivability (protection against one failure). Furthermore, sending a certain amount of data packets less than k on each path should guarantee secrecy against eavesdropping. Thus, the combination of o and l should be in the operational range. Figure 3 is produced for $m = 32$ and coding over $GF(2^8)$.

Figure 4 depicts the required overhead ratio for different PLR when $k = 32, 64$ and 128 and p_{thres}=10^{-12}. Here it is shown that the packet overhead required to keep the PLR below a predefined p_{thres} is less than the required packet overhead to achieve survivability against single link failure.

Figure 5 illustrates the operational range when we consider constraint (9) in addition to the constraints for survivability and secrecy. The $n - m'$ packets sent on the non-failed paths may still be lost due to contentions that means additional overhead is needed. The operational range reduces as the PLR increases. We draw the operational range for p=0.01 and 0.001 besides the one presented in Figure 3.

In the previous Section we choose small parameters for n and k for simplified explanation. We give other examples in Table II where n is equal to $2^q - 1$ and the parameters are chosen to satisfy the survivability and secrecy constraints.

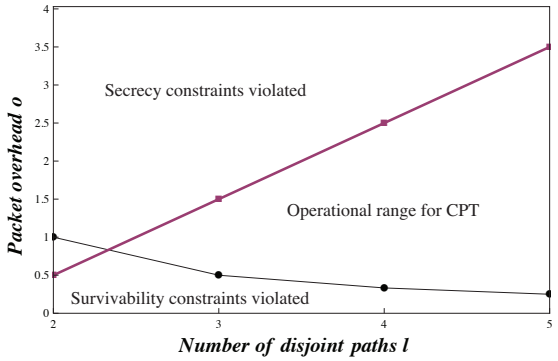


Fig. 3: The operational range for CPT for $m = 32$ and coding over $GF(2^8)$

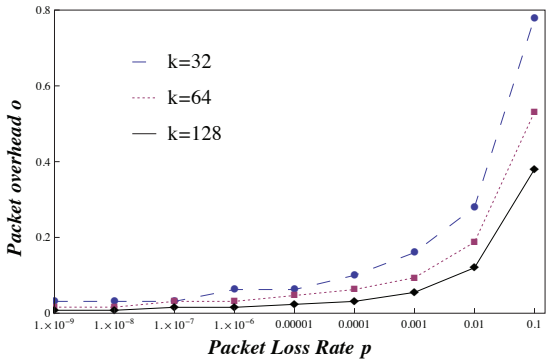


Fig. 4: Required packet overhead for different PLR so that $p_{fail} \leq p_{thres}$ for p_{thres}=10^{-12}

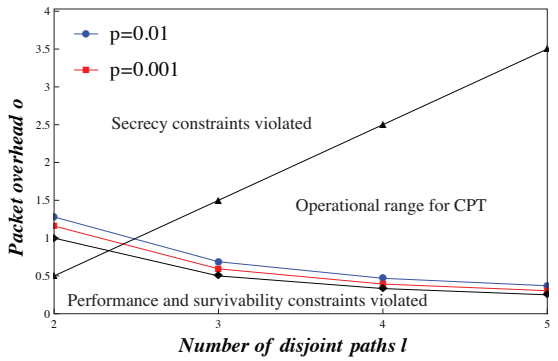


Fig. 5: The reduced operational range for CPT for m=32, coding over $GF(2^8)$ when p=0.01 and 0.001 and p_{thres}=10^{-12}

TABLE II: Examples where $n = 2^q - 1$

q	k	r	o	m'	l	Secrecy level in bits
5	25	6	0.24	5 and 6	6	95
6	42	21	0.5	21	3	126
7	84	43	0.512	42 and 43	3	287
8	204	51	0.25	51	5	1224

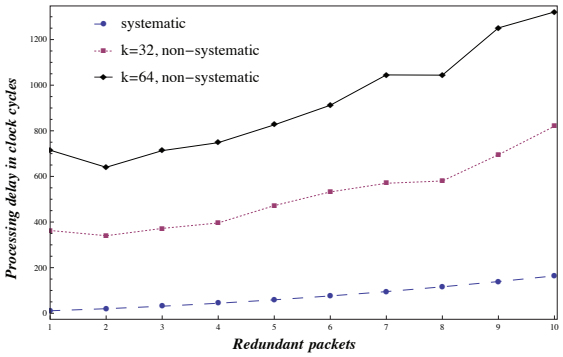


Fig. 6: Processing delay in clock cycles per symbol for a systematic and non-systematic RS code

Figure 6 shows the processing delay for a systematic code and two non-systematic RS codes for $k = 32$ and 64. The processing delay for the non-systematic codes is significantly higher than for the systematic code. This is because that the total number of packets processed by a non-systematic code is greater than the number of redundant packets processed by a systematic code. The processing delay increases as r increases for the both cases. As it is illustrated in Figure 6, the number of data packets k has an impact on the processing delay in addition to r for a non-systematic code. When non-systematic coding is performed, the decoder will always process the packets from the previous packet set while packets from a new packet set arrive. This is not the case for a systematic code for some parameter selections.

We conclude that the survivability, performance and secrecy that the CPT offers are at a price of an increased processing delay. The CPT offers better performance compared to 1+1 or 1+N protection, where the overhead ratio is 1. In order to keep the CPT scheme in the operational range, either the overhead ratio should be greater than 0.5 or the number of disjoint paths should be equal to or more than 3. The CPT scheme offers both survivability and secrecy with less overhead compared to traditional 1+1 path protection, which offers only survivability against a single failure for 100% overhead ratio.

VI. Conclusions

This paper presented the Coded Packet Transport (CPT) scheme, a novel transport mechanism for Optical Packet/Burst Switched (OPS/OBS) networks. The CPT scheme is able to recover packets lost due to contentions and node/link failures, as well as providing secrecy in OPS/OBS networks. We have presented the conceptual architecture for the CPT scheme, along with analytical results, outlining the achievable performance. Further research on this topic should investigate the performance in realistic network topologies.

Acknowledgments

We would like to thank Gergely Biczók for his discussions and remarks that significantly improved the paper.

REFERENCES

[1] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid. Recommendation for key management-part 1: General (revised. In *NIST special publication*. Citeseer, 2006.

[2] G. Biczók and H. Øverby. Combating packet loss in ops networks: A case for network coding. In *NIK 2011*, 2011.

[3] Yuhua Chen and P.K. Verma. Secure optical burst switching: Framework and research directions. *Communications Magazine, IEEE*, 46(8):40–45, 2008.

[4] Damien Giry. BlueKrypt, Cryptographic Key Length Recommendation, 2014.

[5] D. Griffith and SuKyoung Lee. A 1+1 protection architecture for optical burst switched networks. *Selected Areas in Communications, IEEE Journal on*, 21(9):1384–1398, 2003.

[6] D.K. Hunter and Ivan Andonovic. Approaches to optical internet packet switching. *Communications Magazine, IEEE*, 38(9):116–122, Sep 2000.

[7] K. Kralevska, H. Øverby, and D. Gligoroski. Joint balanced source and network coding. In *Telecommunications Forum Telfor (TELFOR), 2014 22nd*, pages 589–592, Nov 2014.

[8] Arjen K Lenstra and Eric R Verheul. Selecting cryptographic key sizes. *Journal of cryptology*, 14(4):255–293, 2001.

[9] Y.-W. Leung. Shared packet loss recovery for internet telephony. *Communications Letters, IEEE*, 9(1):84–86, 2005.

[10] Robert J. McEliece and Dilip V. Sarwate. On sharing secrets and reed-solomon codes. *Communications of the ACM*, 24(9):583–584, 1981.

[11] M. Médard and A. Sprintson. *Network coding, Fundamentals and Applications*. 2012.

[12] Muriel Médard, Douglas Marquis, and Stephen R. Chinn. Attack detection methods for all-optical networks. In *NDSS*. The Internet Society, 1998.

[13] Thinh Nguyen and A. Zakhor. Path diversity with forward error correction (pdf) system for packet switched networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 1, pages 663–672 vol.1, March 2003.

[14] Thinh Nguyen and Avideh Zakhor. Distributed video streaming with forward error correction, 2002.

[15] Paulo F. Oliveira, Luísa Lima, Tiago T. V. Vinhoza, João Barros, and Muriel Médard. Coding for trusted storage in untrusted networks. *IEEE Transactions on Information Forensics and Security*, 7(6):1890–1899, 2012.

[16] H. Øverby, G. Biczk, P. Babarczi, and J. Tapolcai. Cost comparison of 1+1 path protection schemes: A case for coding. In *Proc. IEEE International Conference on Communications (ICC)*, Ottawa, ON, Canada, 2012.

[17] Harald Øverby. Network layer packet redundancy in optical packet switched networks. *Opt. Express*, 12(20):4881–4895, Oct 2004.

[18] Harald Øverby. Traffic modelling of asynchronous bufferless optical packet switched networks. *Computer Communications*, 30(6):1229 – 1243, 2007.

[19] Harald Øverby. Traffic models for slotted optical packet switched networks. *Photonic Network Communications*, 13(2):183–194, 2007.

[20] Harald Øverby. Combined study on survivability and performance in optical packet switched networks. *J. Opt. Netw.*, 7(4):294–309, Apr 2008.

[21] XILINX Product Guide. *LogiCORE IP Reed-Solomon Decoder v9.0*.

[22] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, June 1960.

[23] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[24] Rodney S. Tucker. Optical packet-switched wdm networks – a cost and energy perspective. In *Optical Fiber Communication Conference/National Fiber Optic Engineers Conference*, page OMG1. Optical Society of America, 2008.

[25] V.M. Vokkarane and Qiong Zhang. Forward redundancy: a loss recovery mechanism for optical burst-switched networks. In *Wireless and Optical Communications Networks, 2006 IFIP International Conference on*, pages 5 pp.–5, 2006.

[26] Shun Yao, S. J. Ben Yoo, and B. Mukherjee. A comparison study between slotted and unslotted all-optical packet-switched network with priority-based routing. In *Optical Fiber Communication Conference and Exhibit, 2001. OFC 2001*, volume 2, pages TuK2–TuK2, 2001.

[27] Shun Yao, B. Mukherjee, S.J.B. Yoo, and S. Dixit. A unified study of contention-resolution schemes in optical packet-switched networks. *Lightwave Technology, Journal of*, 21(3):672–683, 2003.