Technische Universität Dresden

Zero-padding Network Coding and Compressed Sensing for Optimized Packets Transmission

Maroua Taghouti

der Fakultät Elektrotechnik und Informationstechnik der Technische Universität Dresden

zur Erlangung des akademischen Grades

Doktoringenieur

(Dr.-Ing.)

genehmigte Dissertation

Vorsitzender:	Prof. Dr.-Ing. habil. Christian Georg Mayr		
Gutachter:	Prof. Dr.-Ing. Dr. h.c. Frank H. P. Fitzek	Tag der Einreichung:	06.04.2021
	Prof. Dr.-Ing. Wolfgang Kellerer	Tag der Verteidigung:	08.11.2021
	Dr.-Ing. Dirk Wübben		

Abstract

Ubiquitous Internet of Things (IoT) is destined to connect everybody and everything on a never-before-seen scale. Such networks, however, have to tackle the inherent issues created by the presence of very heterogeneous data transmissions over the same shared network. This very diverse communication, in turn, produces network packets of various sizes ranging from very small sensory readings to comparatively humongous video frames. Such a massive amount of data itself, as in the case of sensory networks, is also continuously captured at varying rates and contributes to increasing the load on the network itself, which could hinder transmission efficiency. However, they also open up possibilities to exploit various correlations in the transmitted data due to their sheer number. Reductions based on this also enable the networks to keep up with the new wave of big data-driven communications by simply investing in the promotion of select techniques that efficiently utilize the resources of the communication systems.

One of the solutions to tackle the erroneous transmission of data employs linear coding techniques, which are ill-equipped to handle the processing of packets with differing sizes. Random Linear Network Coding (RLNC), for instance, generates unreasonable amounts of padding overhead to compensate for the different message lengths, thereby suppressing the pervasive benefits of the coding itself. We propose a set of approaches that overcome such issues, while also reducing the decoding delays at the same time. Specifically, we introduce and elaborate on the concept of macro-symbols and the design of different coding schemes. Due to the heterogeneity of the packet sizes, our progressive shortening scheme is the first RLNC-based approach that generates and recodes unequal-sized coded packets. Another of our solutions is deterministic shifting that reduces the overall number of transmitted packets. Moreover, the RaSOR scheme employs coding using XORing operations on shifted packets, without the need for coding coefficients, thus favoring linear encoding and decoding complexities.

Another facet of IoT applications can be found in sensory data known to be highly correlated, where compressed sensing is a potential approach to reduce the overall transmissions. In such scenarios, network coding can also help. Our proposed joint compressed sensing and real network coding design fully exploit the correlations in cluster-based wireless sensor networks, such as the ones advocated by Industry 4.0. This design focused on performing one-step decoding to reduce the computational complexities and delays of the reconstruction process at the receiver and investigates the effectiveness of combined compressed sensing and network coding.

Acknowledgements

My Ph.D. journey holds a very special place in my life, which has allowed me to add my name to the truly long list of prodigious researchers who have contributed to the corpus of scientific knowledge. I am glad to be part of this community and feel that I finally accomplished a crucial mission during my passage on planet Earth. It is very humbling to thank all those who have guided and supported me throughout and beyond this expedition. Without them, this achievement would not have been possible.

I would like to start by expressing my deepest gratitude to Prof. Dr.-Ing. Dr. h.c. Frank H. P. Fitzek for giving me the opportunity to be one of his Ph.D. students and for making it possible to be part of the Deutsche Telekom Chair for Communication Networks (ComNets) team. I will always look up to him and will consider him as a reference in my career. I will always be grateful to my mentor, Assoc. Prof. Daniel Enrique Lucani Rötter from Aarhus University, Denmark, as well, who has been one of the first to believe in my small theories and concepts, and who stood by my side during the development of my research ideas. I very much enjoyed his friendly demeanor and specialized "tools of supervision", especially the famous wooden stick he used during my visits that he referred to as the "performance booster" (though it only worked well while I was physically at Aalborg University). I appreciate his patience and understanding throughout all these years and really hope our collaboration and friendship will continue to evolve in the future.

I thank my husband and ex-colleague, Dr.-Ing. Máté Tömösközi, my catalyst who has brought a great balance in my life, for his unconditional love and support. I will never forget when after checking my dissertation for typos, he wrote "vegan" as a comment when he saw that I mistakenly wrote "meat" instead of "meet".

I thank our publishing guru, Prof. Martin Reisslein from Arizona State University, USA, who is not only an exceptional researcher but a one-of-a-kind writer that reminds me of Stephen King. I also thank my other co-authors, especially Dr.-Ing. Anil Kumar Chorppath, Dr.-Ing. Morten Pedersen, Tobias Waurick, Malte Höweler, and Huanzhuo Wu.

I would like to thank my engineering diploma and master's advisor, Prof. Ammar Bouallegue from the National Engineering School of Tunis, Tunisia, and Prof. Ali Abdennadher from Tunisia Polytechnic School of Tunisia, for encouraging me to follow my dreams for seeking a wider knowledge and for pushing me toward the pursuit of my studies abroad. Many thanks go to the people I met during my Ph.D. journey, with a special thanks to Ms. Kirsten Nielsen from Aalborg University, who hosted me in her house in Aalborg, my friends from ComNets, and the Communication Systems Laboratory of the National Engineering School of Tunis, Tunisia.

I, of course, thank my family, especially my sister Dr.-Ing. Myriam Taghouti and her kids, Mrimima and Harun, for the great fun and huge support they were constantly giving to their “tia”. Last, but not least, I will always be grateful to every single person who taught me any new thing during my studies, starting from my aunt, Sabeh Hammami, my kindergarten teacher and high school math teacher, Mr. Hussin-Hadj Sassi, who inspired me to develop my logical thinking, may he rest in peace.

Dedication

To my mother Fatma-Zohra Hammami.

*“C’est par la logique qu’on démontre, c’est par l’intuition qu’on invente.
Savoir critiquer est bon, savoir créer est mieux.”*

Science et Méthodes
Jules Henri Poincaré
1908

Table of Contents

List of Figures

List of Tables

List of Acronyms

5G	Fifth Generation of mobile communication system
AMP	Approximate Message Passing
AONC	Adaptive Opportunistic Network Coding
API	Application Programming Interface
ARQ	Automatic Repeat reQuest
AWGN	Additive White Gaussian Noise
BATS	BATched Sparse code
BIHT	Binary Iterative Hard Thresholding
BP	Basis Pursuit
BPDN	Basis Pursuit De-Noising
BSS	Blind Source Separation
CAIDA	Centre for Applied Internet Data Analaysis
CDF	Cumulative Distributive Function
CdICA	Component-dependent Independent Component Analysis
CF	Chain and Fragment
CH	Cluster Head
CIF	Common Intermediate Format

CNC	Compressive Network Coding
ComNetsEmu	Communication Networks Emulator
CoSaMP	Compressive Sampling Matching Pursuit
CSS	Compressible Source Separation
DCS	Distributed Compressed Sensing
DCS-SOMP	DCS-Simultaneous Orthogonal Matching Pursuit
DCT	Discrete Cosine Transformation
DSC	Distributed Source Coding
ECG	ElectroCardioGram
EEG	Electroencephalogram
EMBP	Expectation Maximization Belief Propagation
EV	Encoding Vector
F2OMP	Finite Field Orthogonal Matching Pursuit
FastICA	Fast Independent Component Analysis
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FHD	Full High Definition
FIT	Fast Iterative Thresholding
FlexONC	Flexible and Opportunistic Network Coding
FNDS	Factory Noise Data Set
FPS	Frame Per Second
FS	Fragmentation Size

GF	Galois Field
GIDNC	Generalized Instantly Decodable Network Coding
GoP	Group of Pictures
GPSR	Gradient Projection for Sparse Reconstruction
HEVC	High Efficiency Video Coding
ICA	Independent Component Analysis
IDNC	Instantly Decodable Network Coding
IHT	Iterative Hard Thresholding
IIoT	Industrial Internet of Things
IoT	Internet of Things
IP	Internet Protocol
IST	Iterative Soft Thresholding
JoComCo	Joint Compress and Code
JPEG	Joint Photographic Experts Group
JSM	Joint Sparsity Model
LARS	Least Angle Regression
LASSO	Least Absolute Shrinkage Operator
LDPC	Low Density Parity Check
LT	Luby Transform
MEC	Mobile Edge Cloud
MIMO	Multiple Input Multiple Output
MMV	Multiple Measurement Vector

MOD	Method of Optimal Directions
MRI	Magnetic Resonance Imaging
MS	Macro Symbol
MSE	Mean Squared Error
MTU	Maximum Transmission Unit
NS3	Network Simulator 3
NSP	Null Space Property
OMP	Orthogonal Matching Pursuit
OSGA	One-Step Greedy Algorithm
OSI	Open Systems Interconnection
P2P	Peer-to-Peer
PRNC	Pseudo-Random Network Coding
PRNG	Pseudo-Random Number Generator
QoE	Quality of Experience
RaSOR	Random Shift and XOR
RE	Restricted Eigenvalue
ReC	Revolving Code
RIP	Restricted Isometry Property
RLNC	Random Linear Network Coding
RoHCv2	Robust Header Compression version 2
ROMP	Regularized Orthogonal Matching Pursuit
RSNR	Reconstruction Signal-to-Noise Ratio

SDN	Software-Defined Networking
S-IDNC	Strict Instantly Decodable Network Coding
SL0	Smoothed l_0
SNC	Sparse Network Coding
SNR	Signal-to-Noise Ratio
SOMP	Smoothed Orthogonal Matching Pursuit
SP	Subspace Pursuit
SSAC	Small Set of Allowed Coefficients
SVC	Scalable Video Coding
SVD	Singular Value Decomposition
TCP	Transport Control Protocol
TP	Trivial Pursuit
TSNC	Tunable Sparse Network Coding
UDP	User Datagram Protocol
URLLC	Ultra-Reliable Low-Latency Communication
VBR	Variable Bit Rate
VNI	Visual Networking Index
WBAN	Wireless Body Area Network
WSN	Wireless Sensor Network

List of Symbols

Network Coding

$\alpha_{\eta n}$	Coding coefficient for source packet n when generating coded packet η
$\mathcal{C}$	Number of macro-symbols that could be XORed together in the RaSOR scheme
$c_{\eta\lambda}$	Coded MS in coded packet η in MS position λ
χ_i	Number of possibly lost coded packets (among the first N transmissions, i.e. non-redundant packets)
C_i	Coded packet of index i
c_{ij}	Coded macro-symbol number j of coded packet C_i
Δ_λ	Source MS degree of column λ, i.e., # of MSs in column λ position $= N - \sum_{\ell=1}^{\lambda-1} \Pi_\ell$
$\Delta_{\max}^{\text{det}}$	Maximum source macro-symbol degree for the deterministic shifting scheme
$\Delta_{\max}^{\text{rand}}$	Maximum source macro-symbol degree for the random shifting scheme
$\Delta_{\max}^{\text{rand/det}}$	Maximum source macro-symbol degree for the random or deterministic shifting scheme
Δ_{opt}	Maximum source macro-symbol degree for the RaSOR scheme
E	payload delivery efficiency
ϵ	packet loss probability
F_S	Fragmentation size, $F_S \leqslant L_{\max}$
K	Number of coded packets sent

λ	MS position, i.e., column position in a given packet, $\lambda = 1, 2, \ldots, \Lambda_{\max}$
Λ_n	Size of source packet i [in MSs], $\Lambda_i = \lceil L_i/\mu \rceil$
$\Lambda_{\max}$	Size of largest source packet in a generation [in MSs]
L_i	Size of source packet i in number of symbols in $GF(2^q)$
ℓ	Number of hops in a line network
$L_{\max}$	Maximum size of source packetin number of symbols in $GF(2^q)$ of a generation
μ	Macro-symbol size in number of symbols in $\mathrm{GF}(q)$
N	Generation size in number of source packets
N_{CF}	Generation size in the *Chain and Fragmentation* scheme
$N_{\max}$	Generation size [in number of source packets] with packet index $n = 1, 2, \cdots, j$, where j corresponds to the maximum number of packets in an IP flow from a CAIDA trace, $N_{\max} \geqslant N$
N_{SB}	Generation size in the *Simple Bundling* scheme
O_G	Ratio of padding overhead of a generation G
$\mathbf{\Omega}$	Real network coding matrix in $\mathbb{R}^{l \times S}$
ω	Window size
O_{MS}	Ratio of padding overhead of a generation G in MSs
$O_{MS,N}$	Ratio of padding overhead of a generation G created due to the macro-symbol design
Φ_η	Size of coded pkt. η, $\eta = 1, 2, \ldots, K$, in MSs
ϕ_i	Random offset to the right of original packet P_i, $\phi_i = 0, \cdots, \Lambda_{\max}$
P_i	Source packet of index i
Π_λ	Source packet size distribution, $\Pi_\lambda = \#$ of source packets with size λ MSs; $\sum_{1 \leq \lambda \leq \Lambda_{\max}} \Pi_\lambda = N$
Ψ	number of full-length packets to transmit in progressive shortening scheme
q	Specification of finite field size
R	Code rate for RLNC, $\frac{N}{K}$
s_{ij}	Source macro-symbol number j of source packet P_i
$s_{n\lambda}$	Source MS in column pos. λ, $\lambda = 1, 2, \ldots, \Lambda_n$ in source packet n
υ_{RLNC}	Overhead due to coding coefficient
χ_i	Number of lost coded packets

Compressed Sensing

$\mathbf{A}$	Measurement matrix in $\mathbb{R}^{m\times n}$, incorporates $\mathbf{\Psi}$ and $\mathbf{\Phi}$
$\mathbf{a_i}$	i-th raw of $\mathbf{A}$
$\mathbf{D}$	Over-complete dictionary in $\mathbb{R}^{m\times n}$ (change dim)
k	Sparsity
n	Length of signal $\mathbf{x}$
$\mathbf{\Phi}$	Measurement matrix in $\mathbb{R}^{n\times m}$
P_{rx}	Probability that ensures the delivery of at least $l_j - 1$ measurements to CH$_j$
$\mathbf{\Psi}$	Transformation matrix in $\mathbb{R}^{n\times n}$
$\mathbf{\Psi_T}$	Projection matrix in the temporal sparse basis, in $\mathbb{R}^{n\times n}$
Q	Separation quality for CSS
S	Number of sensors in a cluster
θ	Sparse coefficient vector in $\mathbb{R}^n$
$\mathbf{\Theta}$	Sparse representation of $\mathbf{X}$ in $\mathbb{R}^{S\times n}$
ε	Link loss rate
$\mathbf{x}$	Original data vector in $\mathbb{R}^n$
$\mathbf{X}$	Original readings from S sensors, $\mathbf{X} = [\mathbf{x_1} \cdots \mathbf{x_S}]$ in $\mathbb{R}^{S\times n}$
$\hat{\mathbf{x}}$	Estimated reconstructed signal in $\mathbb{R}^n$
$\mathbf{x_i}$	Original data vector in $\mathbb{R}^n$
$\mathbf{x_{ji}}$	Original data vector of sensor node i in cluster j
$\mathbf{y}$	Measurement data vector in $\mathbb{R}^m$
$\mathbf{Y}$	Temporally compressed readings, $\mathbf{Y} = [\mathbf{Y_1} \cdots \mathbf{Y_S}]$ in $\mathbb{R}^{S\times m}$
$\mathbf{y_i}$	Measurement data vector in $\mathbb{R}^m$
$\mathbf{y_i}$	Temporally compressed data vector of sensor node i in cluster j
$\mathbf{Z}$	Data from all cluster heads after spatial compression
$\mathbf{z_c}$	Common component
$\mathbf{z_i}$	Innovative component
$\mathbf{Z_j}$	Data from cluster head j after spatial compression

Chapter 1 | Introduction

This chapter highlights the current technological trends and requirements where our contributions throughout this dissertation are a potential fit. We discuss our motivations based on the latest advances, focusing on the current trend in the data deluge problem, which is partially due to data provided from sensors deployed in all of the IoT devices. Our vision in curbing such an issue relies on the adoption of key 5G innovation techniques, that promote either the data size reduction, such as compressed sensing techniques, or the overhead carried in the network and later stored in devices, using the network coding techniques. Furthermore, we highlight our main contributions of this dissertation and present a collection of our published research papers that support our theses. We also include a roadmap to guide readers to easily navigate between independent topics in the following chapters.

1.1 Context and Motivations

We highlight the enormous impact of Big Data on current 5G communication systems and relate its inter-connection with IoT, namely the sensors. We discuss the continuum of data deluge and the newly established inter-connections between sensors, IoT, and Big Data in the new era of 5G. Furthermore, we introduce the key innovation techniques adopted in this dissertation to reduce the data volume and the overhead needed for the transmission.

1.1.1 Data Deluge

The IoT is considered to be the fastest emerging technological trend since it provides a limitless supply of information that has never been witnessed before [15]. Sensors are an integral component of IoT that are embedded everywhere. They are small and power-constrained devices capable of detecting and responding to any environmental change. The commonly deployed IoT sensors record temperature, humidity, infrared, gyroscope, accelerometer, proximity, and optical readings, to name a few. They are crucial for enabling intelligent road traffic monitoring, water pipe leakage monitoring, congestion avoidance, smart cities establishment [16], etc. This universal deployment of sensors results in the generation of a massive amount of sensory data [17]. Additionally, the current number of IoT devices is derisory compared to what is expected in the next decade. The democratization of connected devices with the advent of 5G are the core drivers of this explosion. This pervasive connectivity will lead to having an estimated 10^7 devices per km^2 in dense areas and over 125 billion devices by 2030 [18, 19].

As data has become a fundamental factor of production, having over 50 billion sensors deployed throughout all IoT devices, which instantly measure various physical phenomena and other types of signals, allows for continuous monitoring and human-less decision making in a wide range of scenarios in conjunction with Industry 4.0. In many cases, this data is collected and later analyzed to create certain statistics about the overall performance and surrounding conditions, such as the factory floor temperature and humidity to detect anomalies or defect devices. Its requirements have further stress-tested our ability to do fast data ingestion and quickly discover insights with minimal delay for real-time scenarios such as monitoring. IoT sensors have the critical role of improving the industrial sector efficiency by decreasing the overall costs, as well as increasing workers' safety. For instance, it has become crucial to monitor industrial IoT devices on a factory floor in order to remotely detect any anomalies or deficiencies in the responsible device. The deployed sensors are therefore continuously transmitting the acoustic data to remote servers, which in turn, are capable of detecting any occurring problem at an early stage problem detection, thus preventing the production breakdown. Nevertheless, with the staggering number of sensors that generate massive data, it has become a challenging task to quickly and accurately locate malfunctioning devices. Additionally, the high latency during remote data separation could yield to a poor Quality of Experience (QoE). These IoT devices do not just generate environmentally sensed data, but also a huge portion is just video provided from high-end devices such as smartphones, 360-degree video, and variable video quality from drones and surveillance cameras broadly deployed to assist 5G communication systems. By 2021, the volume of mobile video delivery is expected to reach 78% of the mobile data traffic [20].

One of the major side effects of IoT deployments is that they generate a massive amount of data that hugely contributed to the big bang of big data. And despite the fact that they evolved independently, IoT and big data have become inevitably inter-connected or interrelated over time. Generally, the collected sensory data is labeled as "data rich, information poor", which could be translated into correlations or redundancy that could be intelligently dealt with, instead of increasing the demands on storage systems. As a matter of fact, data can be spatially, temporally, or dynamically correlated (combined types of correlations), as well as structured or unstructured. Information extracted from data could differ in complexity, provenance, reliability, representation, etc. It can also differ in the rate or pace at which it is generated and accessed.

On the other hand, in order to avoid confusion with the term big data from other fields perspectives, we remind the reader that the data classified as big data is checked using the concept of five V, using health data;

i) *Volume*: there is a massive volume of data created by hospitals and clinics that are estimated at around 2314 exabytes annually.

ii) *Velocity*: the data is usually generated at a high-speed

iii) *Variety*: there exist various types of data; structured, e.g., Excel files, semi-structured, e.g., log records, and unstructured, e.g., X-Ray images.

iv) *Veracity*: the data must be accurate and trustworthy.

v) *Value*: allows for faster disease detection and more adequate treatments at reduced costs.

This definition matches perfectly the data that the sensors continuously generate. The increasing challenges in managing Big Data are inciting for revolutionary and fundamental techniques, as well as for technologies that are expected to handle the complexity and the unprecedented volume of this data. Therefore, from our perspective, it is important to focus on the root problem, namely *sensors lead to big data*. This dissertation does not propose solutions and techniques to analyze and manage large amounts of data. Instead, it presents means to reduce the data volume from the perspective of the communications field. We strongly believe that common wisdom and standard analysis are no longer sufficient to face this unprecedented increase in the staggering volume of data, and more efforts should be invested at the early stages of the data cycle to curb its impact on communications and storage systems.

1.1.2 Key Innovation Techniques

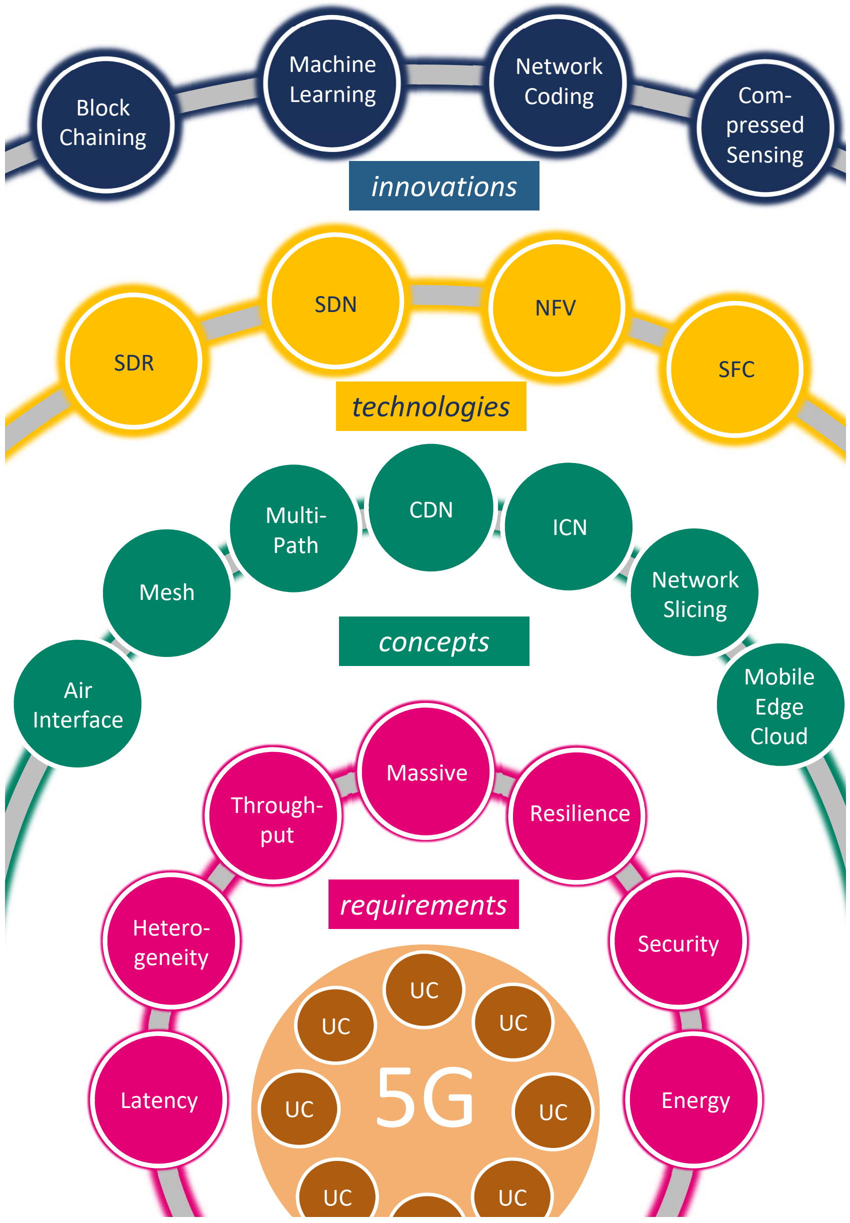


Figure 1.1: The 5G atom [1]

Globally, IP traffic will reach an annual run rate of 2.3 Zettabytes in 2020, up from 870.3 Exabytes in 2015 [20, 21]. This has urged the need for proposing innovative approaches to deal with this amount of data, which consists mainly of video and environmentally sensed data, in all stages of the data cycle. The METIS project proposed to use network coding in the 5G standard [22]. Figure 1.1 illustrates the 5G atom that describes the vision at the Deutsche Telekom Chair of Communication Networks, where most of the research related to this dissertation was conducted, for the main attributes and contributors to the 5G standard, mainly the concepts, technologies, and innovations proposed for the 5G use cases and their stringent requirements. The fourth tier of the 5G atom, *innovations*, relies mainly on network coding and compressed sensing, which will be the main focus of this dissertation since the inevitable data increase has led to the convergence of these innovative techniques. Network coding has the outstanding ability to significantly improve the throughput, robustness, transmission efficiency, delay reduction, etc. Compressed sensing proved its ability to reduce the size of any type of data, such as sensor readings, audio, medical imaging, etc., without altering it when it is later reconstructed. It is also known that the reconstruction algorithms for compressed sensing are stable, flexible, as well as scalable, which widens the potential applications for such a technique.

1.1.2.1 Network Coding

Network coding [23] is a coding technique that breaks with the end-to-end view on data dissemination of the old-fashioned store-and-forward networks. Network coding promotes a novel networking paradigm where the intermediate nodes are no longer passive entities that store and forward data packets. Its exclusive in-network computing ability allows for increased overall network performance. More interestingly, it is considered as an enabler of flexible and disruptive future network designs, known to be rather unstructured, very dynamic, and highly heterogeneous, as is the case with IoT systems. Network coding is widely used in 5 G-related applications and critical systems, including Ultra-Reliable Low-Latency Communication (URLLC), industrial robotics, connected vehicles, etc. Its remarkable performance resides in the fact that it is the unique rateless code that allows for recoding at intermediate nodes, unlike legacy coding techniques that would require decoding at relay nodes before encoding again.

The most adopted form of network coding is Random Linear Network Coding (RLNC) [24], [25], when provided a set of coding coefficients drawn at random from a finite field, creates a coded packet using the combination of original packets mapped with these random coefficients. At least as many coded packets as the original packets, along with the coefficients employed, are needed by the receiver to recover the original packets, as it is done by analogy to linear systems of equations. Because it is traffic agnostic, RLNC represents one of the best solutions for wireless mesh networks with varying loss rates.

1.1.2.2 Compressed Sensing

Compressed sensing [26, 27] is the field that enables the solving of under-determined linear systems of equations, provided that the system is very sparse, using optimization algorithms such as convex relaxations thanks to the established equivalence between the ℓ_0 and the ℓ_1 norm [28]. As a result, a signal could be compressed far beyond the Shannon-Nyquist theory. Despite the challenging mathematical background, few properties must be fulfilled in order to guarantee the exact or approximate reconstruction of the compressed data. For example, a non-sparse signal could undergo a sparsification process using well-known transforms such as the Discrete Cosine Transformation (DCT) or simply by designing an over-complete dictionary [29].

In a nutshell, any set of sufficiently large data could be compressed and reconstructed efficiently exactly or approximately. This has widened the applications for compressed sensing and made it one of the most promising innovation technologies for IoT devices known to generate massive amounts of data that cannot be handled easily with the current data management systems. Therefore compressed sensing serves as an effective data communications plugin that reduces the delay of data transmission and the load on the storage systems.

1.2 Key Contributions

This dissertation's contributions consist mainly of proposing new RLNC approaches that minimize the padding overhead when transmitting variable-sized packets. Three major novel techniques were designed, each of which could be used under specific scenarios and conditions. First, we proposed the first RLNC based scheme that generates unequal-sized coded packets regardless of their distributions, when coding on a finer grain using the concept of Macro Symbol (MS) [5] that we introduce in this thesis. Such a scheme called "*Progressive Shortening*" [5] creates coded packets with an optimized percentage of padding overhead. Moreover, we proposed a linear complexity encoding and decoding scheme, called "*Random Shift and XOR (RaSOR)*" [30], which works using only binary XORing operations on macro-symbols, to create less coded packets compared to traditional RLNC, very low padding overhead, very low encoding vector overhead, and, obviously, very low computational complexities for encoding and decoding. Finally, we advocated for a more flexible scheme, called the "*Deterministic Shifting*" [4] scheme that fully exploits the advantages of RLNC while producing less coded packets and less padding overhead, thanks to the accurate design of the macro-symbols and the shifts that the uncoded packets undergo during the precoding step.

On the other hand, we proposed the first compressed sensing and network coding scheme "*Joint Compress and Code (JoComCo)*" [8] for Wireless Sensor Network (WSN) that judiciously combines both techniques while preserving their major features, such as the robustness, resilience, and the ability to recode for network coding, as well as the correlations exploitation to recode the data size for compressed sensing. We also investigate the right parameters to maximize the gains from this joint design. For instance, our numerical results show that real network coding normalized coefficients drawn from the Gaussian distribution do not alter the compression gains and allow for higher reconstruction Signal-to-Noise Ratio (SNR). Furthermore, in Appendix B, we contributed with an implementation of compressed sensing for simple topologies to give the students a flavor about the feasibility of this technique in the communication field [11].

1.3 Organization of the Thesis

All the topics covered in this dissertation were originally published at peer-reviewed international conferences and journals. This means that the results discussed, including equations, figures, etc., were extracted from our publications, unmodified or slightly modified so that they match the unified writing style we designed for this dissertation. This clearly affects in part the text, because rewriting everything knowing that the concepts, ideas, results, etc., are the same, is a very challenging task. We also deduce from our experience in writing this manuscript, that avoiding self-plagiarism could lead to diverging from the straightforward meaning of the specific topic discussed. We tried to make it clear at the right positions of the extracted text where it was originally published. Furthermore, we note that some of the results present in this manuscript were not previously published, but we are soon submitting them for review in some journals and conferences.

We briefly introduce the different topics covered in the dissertation, chapter by chapter, in order to facilitate the navigation through it. As the wide lines of the dissertation are mainly related to the techniques of network coding and compressed sensing, the reader could skip some chapters without affecting the reading chronological order.

Chapter 2 provides a survey-style overview of the theories of network coding and compressed sensing. The first part deals with RLNC more specifically. It reviews the major contributions related to RLNC, including the variant approaches of RLNC, as well as the main their main characterizations in terms of overhead, computational complexities, and decoding delays, i.e. probability of decoding, based on different scenarios and topologies. The second part explains the crucial properties to have a compressed sensing problem as well as accurately solving it. It also presents the results of our combined design of compressed sensing and blind source separation for industrial applications. Chapter 3 unveils the randomness in the distributions of packet sizes of various real-life traces, and how most of the coding techniques, including RLNC, deals naively with it using the zero-padding approach when coding packets with variable sizes. It characterizes its magnitude and impact on the coded packets to transmit, using different traces from Internet Protocol (IP) core networks (TCP and User Datagram Protocol (UDP)), and the standard video codecs, such as VP9 and H.264. Additionally, it studies the existing approaches to overcome the issue of the zero-padding and specifies the percentage of the remaining padding overhead, as well as the exact mathematical expressions for the computational complexities. Chapter 4 explains the concept of the macro-symbols, which are subsets of the packet in the operating finite field. Additionally, it presents the first macro-symbol-based schemes, mainly the *Progressive shortening* approach, and it compares its performance to the standard RLNC. Chapter 5 introduces different macro-symbol approaches, which rely on right shifting the macro-symbols before performing the coding operations. It provides a full analysis of their decoding delay and compares the performance of all the state-of-the-art approaches and the ones advocated in this dissertation using video traces. Chapter 6 describes the efficient design of an in-network computing scheme based on the combination of network coding and compressed sensing for WSN. Finally, Chapter 7 gives a summary of the main contributions and draws the conclusions of this thesis. Furthermore, it suggests some research directions that could be carried out in the future. It is worth mentioning that Chapter 1- 2 serve as a general introduction and overview of the topics covered in the dissertation. Chapters 3- 5 are inter-connected, and they cover the problem of zero-padding when transmitting variable size packets using the technique of RLNC. Chapter 6 however, stands on its own and could be studied separately from the aforementioned chapters. Figure 1.2 summarizes the key contributions of the dissertation.

1.4 Original Publications

1. [3] Maroua Taghouti, Daniel E. Lucani, Morten V. Pedersen, and Ammar Bouallegue. "On the impact of zero-padding in network coding efficiency with Internet

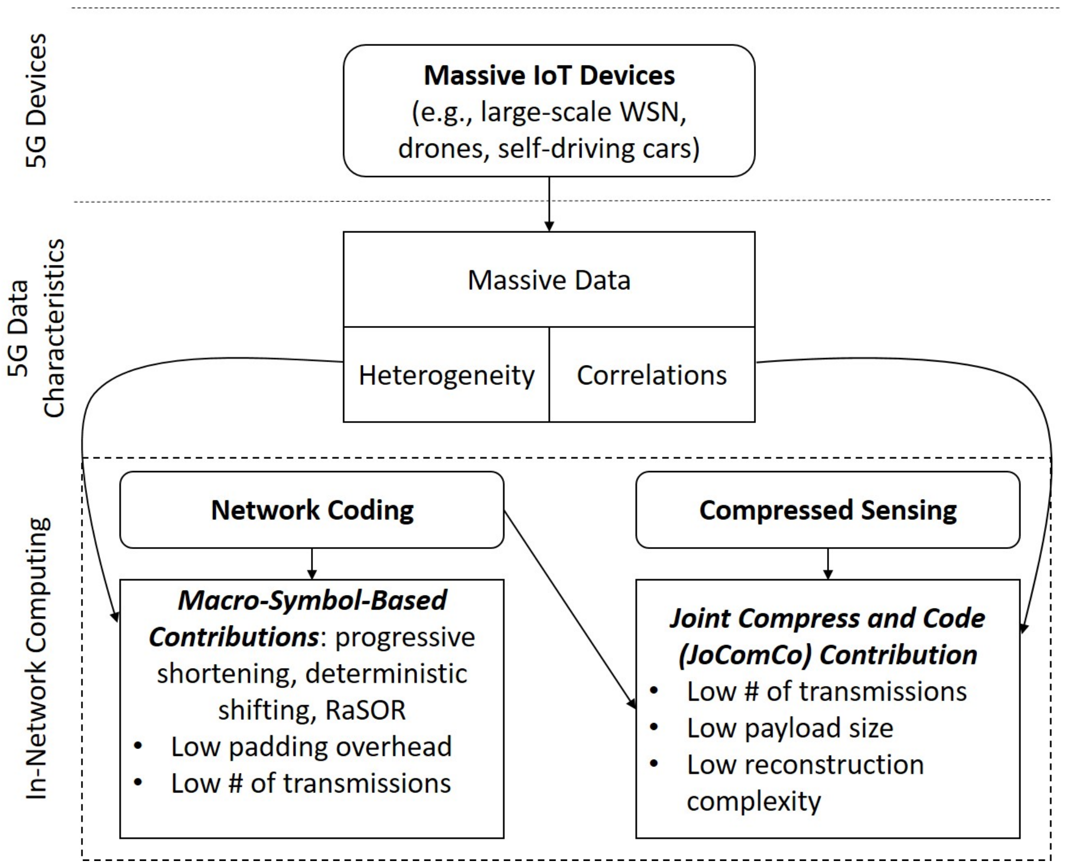


Figure 1.2: General overview of the key contributions.

traffic and video traces". in 22nd European Wireless Conference, (EW), 2016.

2. [5] Maroua Taghouti, Daniel E. Lucani, Morten V. Pedersen, and Ammar Bouallegue. "Random Linear Network Coding for streams with unequally sized packets: Overhead reduction without zero-padded schemes". in 23rd IEEE International Conference in Telecommunications, (ICT), 2016.

3. [30] Maroua Taghouti, Daniel E. Lucani, and Frank H. P. Fitzek. "Random Shift and XOR of unequal-sized packets (RaSOR) to shave off transmission overhead". in 51st IEEE Annual Conference on Information Sciences and Systems, (CISS), 2017.

4. [31] Maroua Taghouti, Daniel E. Lucani, Frank H. P. Fitzek, and Ammar Bouallegue. "Random Linear Network Coding schemes for reduced zero-padding overhead: Complexity and overhead analysis". in 23rd European Wireless Conference, (EW), 2017.

5. [6] Maroua Taghouti, Mate Tömösközi, Malte Howeler, Daniel E Lucani, Frank

H. P. Fitzek, Ammar Bouallegue, Peter Ekler "Implementation of network coding with recoding for unequal-sized and header compressed traffic". in IEEE Wireless Communications and networking conference (WCNC), 2019

6. [4] Maroua Taghouti, Daniel E Lucani, Juan A. Cabrera, Martin Reisslein, Morten Videbæk Pedersen, and Frank H. P. Fitzek, "Reduction of padding overhead for RLNC media distribution with variable size packets". in IEEE Transactions on Broadcasting, 2019

7. [2] Maroua Taghouti, "Compressed Sensing", book chapter, Fitzek, Frank H P; Granelli, Fabrizio ; Seeling, Patrick (Ed.): 1 , Chapter 10, Elsevier, 1, 2020

8. [11] Maroua Taghouti, and Malte Howeler "In-network Compressed Sensing", book chapter, Fitzek, Frank H P; Granelli, Fabrizio ; Seeling, Patrick (Ed.): 1 , Chapter 22, Elsevier, 1, 2020

9. [10] Maroua Taghouti, Tobias Waurick, Mate Tömösközi, Anil K. Chorppath, and Frank H. P. Fitzek, "On the Joint Design of Compressed Sensing and Network Coding for Wireless Communications". in Transactions on Emerging Telecommunications Technologies, 2019

10. [8] Maroua Taghouti, Tobias Waurick, Anil K. Chorppath, and Frank H. P. Fitzek, "Practical Compressed Sensing and Network Coding for Intelligent Distributed Communication Networks". In 14th International Wireless Communications and Mobile Computing Conference (IWCMC), 2018

11. [9] Maroua Taghouti, Tobias Waurick, Anil K. Chorppath, and Frank H. P. Fitzek, "On the Design of a Joint Compressed Sensing and Network Coding Framework". In 24th European Wireless Conference (EW), 2018

12. [32] Ludwig, S., M. Karrenbauer, A. Fellan, H. D. Schotten, H. Buhr, S. Seetaraman, N. Niebert, A. Bernardy, V. Seelmann, V. Stich, A. Hoell, C. Stimming, H. Wu, S. Wunderlich, M. Taghouti, F. Fitzek, C. Pallasch, N. Hoffmann, W. Herfs, E. Eberhardt, and T. Schild- knecht "A 5G Architecture for the Factory of the Future"

13. [33] Huanzhuo Wu, Ievgani Tsokalo, Maroua Taghouti, Hani Salah, and Frank H. P. Fitzek, "Compressible Source Separation in Industrial IoT Broadband Communication". in 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2019

14. [34] Karrenbauer, M., S. Ludwig, H. Buhr, H. Klessig, A. Bernardy, H. Wu, C. Pallasch, A. Fellan, N. Hoffmann, V. Seelmann, M. Taghouti, et al. (2019) "Future industrial networking: from use cases to wireless technologies to a flexible system architecture," at-Automatisierungstechnik, 67(7), pp. 526–544

15. [35] M. Karrenbauer et al., "Towards a Flexible Architecture for Industrial Networking," 23th VDE/ITG Conference on Mobile Communication (23. VDE/ITG Fachtagung Mobilkommunikation), Osnabrück, 2018.

16. [36] H. Klessing, S. Ludwig, M. Karrenbauer, H. Schotten, H. Wu, M. Taghouti, F. H. P. Fitzek, and P. T. Lozano, "A Factory of the Future Reference Network Architecture", submitted to IEEE Communications Magazine.

17. [37] Maroua Taghouti, "PhD Forum: Padding Overhead Reduction in Random Linear Coded Variable Size Media Streams", 22nd IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2021

1.5 Notations and General Terms

In order to clarify the mathematical notations used, we note in the remainder of the dissertation that bold lower case symbols represent one-dimensional vectors and bold upper case symbols represent two-dimensional vectors, i.e., matrices. Additionally, $[\![a\ ;\ b]\!]$ denotes intervals of integers, which is equivalent to $\{a, a+1, \cdots, b-1, b\}$. The ceiling function which gives the smallest integer larger or equal to a is defined by $\lceil a \rceil$. We refer to the rank of a matrix as rk, whereas $rk\{\mathbf{a_1}, \cdots, \mathbf{a_n}\}$ represents the rank of vectors $\mathbf{a_1}, \cdots, \mathbf{a_n}$. Finally, we remark that GF(a) and $\mathbb{F}_a$ are used interchangeably.

Chapter 2 | Disruptive Innovation Techniques

Inefficient traditional signal acquisition techniques and the mathematically proved possibility in compressing far beyond the Shanon-Nyquist theory gave rise to the disruptive compressed sensing field. On the other hand, network coding is the only Forward Error Correction (FEC) code that allows for recoding at intermediate nodes, which is extremely important for wireless mesh networks. Both techniques are considered pillar enabler technologies for 5G. This chapter provides an introduction of these key techniques that this dissertation relies on, and provides a complete survey style overview that allows the reader to easily follow the remainder of the dissertation. In particular, it proposes a comprehensive taxonomy of the main concepts and the branches mainly related to the RLNC technique. This study is intended to bring insights to engineers as well as researchers, regarding the performances in terms of decoding delays, i.e. decoding probabilities, computational complexities, and overheads of these RLNC based techniques. Furthermore, it proposes a lightweight overview of compressed sensing as well as the crucial components and properties to be fulfilled to allow any set of data to be efficiently reconstructed [1, 2]. Finally, it proposes our latest results in combining blind source separation and compressed sensing in the context of Industry 4.0 [32, 35, 36].

2.1 Network Coding Theory

Network coding is a disruptive paradigm that was first introduced at the turn of the millennium by Ahlswede et al. [38]. The breakthrough of network coding was that intermediate nodes were able to recode the received data without the need to decode first, unlike all other coding techniques, such as Reed Solomon and Raptor codes.

2.1.1 Network Coding Branches

Looking at the big picture of network coding, we can classify its major directions as illustrated in Figure 2.1, into a Linear and non-linear Network Coding discipline. This represents our vision in classifying the different network branches, thus it is not unique [39]. We note that non-linear and deterministic network coding are the branches with the least research interest given, because of their limited improvements compared to the other branches. Douik et al. provided a comparison between RLNC, opportunistic network coding, and instantly decodable network coding, based on various criteria such as throughput, delay, overhead, buffer size [40].

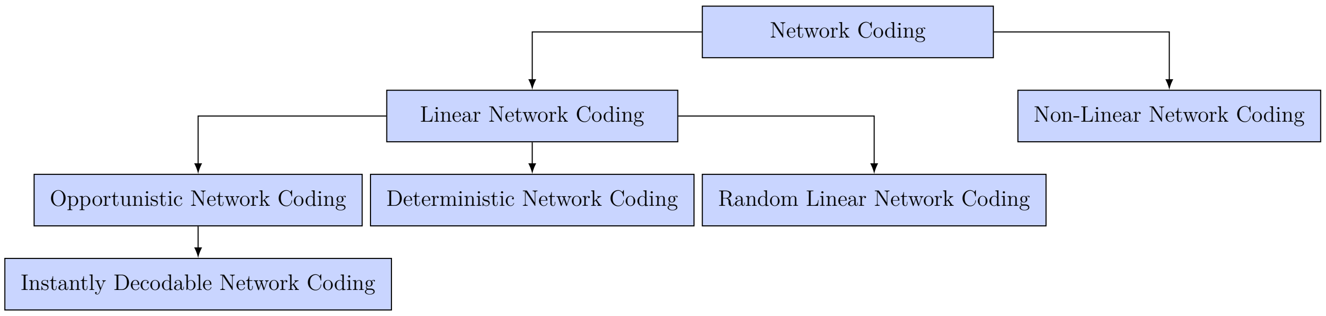

Network Coding
Linear Network Coding
Non-Linear Network Coding
Opportunistic Network Coding
Deterministic Network Coding
Random Linear Network Coding
Instantly Decodable Network Coding

Figure 2.1: The most important branches of Network Coding.

2.1.1.1 Linear Network Coding

It encloses most of the common types of network coding in the literature. It achieves the min-cut max-flow [41, 42]. Two key approaches are found under the linear network coding umbrella, namely inter-flow and intra-flow network coding, also called inter-session and intra-session network coding. The former is performed on packets from different network flows, which complicates the problem in general. That is why opportunistic network coding (discussed below) guarantees some of the common network coding benefits. On the other hand, intra-flow network coding is performed on packets from a unique network flow to increase the resilience to packet losses. Additionally, its recoding ability helps in compensating for lost packets, thus increasing the throughput. There are also research works and implementations that combine inter-flow network coding to efficiently exploit the network topology, and intra-flow network coding to increase the robustness against packet loss, such as the CORE protocol [43, 44]. This diversity of the design and applications of linear network coding has led to the design of the three following sub-branches.

2.1.1.1.1 Opportunistic Network Coding It is a simple technique that allows the intermediate nodes to decide whether to transmit an uncoded packet, if no additional packets are available, or simply wait for the next opportunity to create a coded packet and transmit it. Proposed by Chen et al., it aims at overcoming the packet loss rates and the large delays incurred due to the encoding process of network coding [45]. Opportunistic network coding approaches join the field of inter-flow network coding and opportunistic routing, such as Flexible and Opportunistic Network Coding (FlexONC) and Adaptive Opportunistic Network Coding (AONC). The former [46] creates coding opportunities by considering the packets of the neighboring nodes, while the latter enhances the video transmission quality of wireless multimedia sensor networks [47]. The performance of opportunistic network coding usually depends on the network topology, but overall it has a moderate decoding delay, sub-optimal throughput compared to random network coding, and allows progressive coded packets decoding [40].

2.1.1.1.2 Instantly Decodable Network Coding It is a subclass of the *Opportunistic Network Coding* that was first introduced by Traskov et al. [48]. The encoding is performed using only XOR operations. Additionally, it is characterized by the fact that every received coded packet is immediately employed to decode an original packet without being kept for future decoding opportunities, and the non-instantly decodable ones are simply discarded. This makes the decoding process straightforward and much simpler. Since it performs XOR operations only, Instantly Decodable Network Coding (IDNC) has a low overhead and computational complexity, which results in low decoding delays. However, it has a sub-optimal throughput and its performance is mostly dependent on the feedback, compared to standard RLNC [40]. Overall, it has two major branches, namely Strict Instantly Decodable Network Coding (S-IDNC) [49], where the sender is constrained to either send instantly decodable (systematic) or non-innovative packets, and Generalized Instantly Decodable Network Coding (G-IDNC) [50] that relocates this constrained to the receiver. This logically makes the valid coded packets in S-IDNC valid in GIDNC as well, but not necessarily the other way around [40]. This type of network coding is better applied on critical real-time applications such as video streaming and online video games, where the outdated packets are not needed [51, 52].

2.1.1.1.3 Deterministic Network Coding It uses predefined and unchanged coding coefficients that fulfill the requirements for efficient performance of a certain strategy. Deterministic network coding is employed in static topologies such as wired transmissions, where the channel conditions do not usually change, and can also bring high network coding gains when the network sizes increase.

2.1.1.1.4 Random Linear Network Coding It provides a practical and distributed approach to fully exploit the potential benefits of network coding [25]. In a nutshell, it creates coded packets by randomly combining the packets using randomly and uniformly generated coding coefficients from a finite field. The decoder solves then the resulting linear system of equations and retrieves the original packets [53]. Unlike traditional FEC codes, its ability for recoding, without the need of first decoding, allows it to increase the throughput and the robustness against packet losses. Per contra, it is better suited for scenarios that tolerate relatively large delays, as progressive decoding cannot be supported with the conventional approach. Some RLNC variants were designed to overcome the limitations of the standard RLNC. These are later discussed in this chapter. We note that in this dissertation we are only interested in the RLNC based techniques.

2.1.1.1.5 Non-Linear Network Coding Non-linear network coding tackles the problems related to arbitrary networks, where linear codes cannot achieve the capacity or that do not have a scalar linear solution over any finite field [54], such as Cockroach network, Caterpillar network, and Beetle network [55, 56]. As far as we know, there is very little research interest given to this type of network coding.

2.1.2 Random Linear Network Coding

Chou et al. introduced a practical solution for network coding that obviates the need to learn about the encoding and decoding operations, as well as the topology [57], preparing the playground for its efficient deployment. In order to reduce its large costs, they proposed to perform generation based RLNC, which operates on subsets of the original packets, called *generations*, also referred to as chunks [58], classes [59], batches [60], etc.

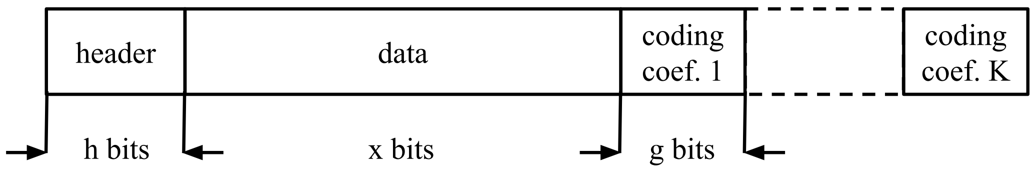


Figure 2.2: Structure of a coded data packet.

According to their design, the resulting coded packets should have a part reserved to the coding coefficients as depicted in Figure 2.2, generally referred to as the coding vector. It reveals all the information needed to learn about the operations performed at the packets. This allows the decoder to recover the original packets by inverting the coding operations. We note that all the operations involved in the process of RLNC encoding/recoding/decoding are performed on finite fields only. In the following, we consider a point-to-point scenario, for simplicity and without loss of generality to a multicast topology.

2.1.2.1 Encoding

We consider transmitting a generation consisting of N input packets $P_1, \cdots, P_N$ to a destination node, without loss of generality to multiple generations transmission, as they are independent, or simply non-overlapping. We suppose that the link is perfect and no erasures or losses could occur. Formally,

$$C_\eta = \sum_{i=1}^{N} \alpha_{\eta i} P_i, \tag{2.1}$$

where $\eta \in [\![1 \; ; \; K]\!]$ represent the index of the created coded packet. The encoding process could be also expressed in terms of matrices as:

$$\begin{bmatrix} \mathbf{C_1} \\ \vdots \\ \mathbf{C}_\eta \\ \vdots \\ \mathbf{C_K} \end{bmatrix} = \underbrace{\begin{bmatrix} \alpha_{11} & \cdots & \alpha_{1N} \\ \vdots & & \vdots \\ \alpha_{\eta 1} & \ddots & \alpha_{\eta N} \\ \vdots & & \vdots \\ \alpha_{K1} & \cdots & \alpha_{KN} \end{bmatrix}}_{\mathbf{M}} \begin{bmatrix} \mathbf{P_1} \\ \vdots \\ \mathbf{P_N} \end{bmatrix}, \tag{2.2}$$

where the coding matrix (also called decoding matrix) $\mathbf{M}$ contains all the coefficients mapped to the original packets, where the number of columns represent the number of original packets P_i and the rows refer to the number of coded packets generated. Since RLNC is a rateless code then K does not have an upper bound, and successful decoding depends only on obtaining exactly N linearly independent (also called innovative) coded packets among the K created ones. Moreover, $\mathbf{M}$ is a dense matrix, because the coefficients are independently and uniformly generated at random from a finite field.

It is a versatile code that does not require any state tracking, and where the code is embedded within the data itself. The coding coefficients used to create a coded packet C_η are stored in the encoding vector so that they are later used by the decoder to retrieve the original packets. Figure 2.2 illustrates where the coding coefficients vector is padded to a coded packet. This incurs the creation of coding vector overhead. It usually depends on the finite field choice as well as the generation size. Let υ_{RLNC} be the coding vector overhead that we express as:

$$\upsilon_{\text{RLNC}} = \log_2(q) \cdot N \quad (bits) \tag{2.3}$$

, where q is the finite field size used. As for the network coding implementation, there exists the Kodo library, which is a friendly network coding framework that contains the fastest finite fields library [61]. This makes the RLNC implementation faster and more optimized.

2.1.2.2 Recoding

It is typically performed by the intermediate nodes (relays), and it consists of the process where the coded packets are re-encoded (recoded) without the need to decode them first. This operation involves both the coded packets (payload), as well as their corresponding encoding vectors. Similar to the encoding process, the intermediate node draws random coefficients from the previously used finite field and combines the coded packets and their coding coefficients too. This operation guarantees to preserve the packet and the coding vector sizes. This is the most important reason why network coding is usually performed using finite fields.

2.1.2.3 Decoding

Since RLNC encoding creates a generation of coded packets as a linear system of equations, decoding is nothing more than solving these equations using the traditional algorithms, mainly using Gaussian elimination. We recall that it requires two steps, namely *triangulation step* and *backward substitution step*. Specifically, it consists of converting the decoding matrix that contains the encoding coefficients using elementary row operations, then column reordering in order to obtain the shape of a triangular matrix. The back substitution step consists of converting this triangular matrix into the identity matrix. We note that decoding is mainly commanded by multiplication operations performed during the Gaussian elimination, which is a leading factor of the decoding computational complexity. In general, the decoding complexity of conventional

RLNC is cubic, and we note it by $\mathcal{O}(N^3)$ in the finite field used. Nonetheless, other approaches that inherit most of the RLNC properties have lower decoding computational complexities, especially those which have sparse coding vectors, as they result in sparse decoding matrices, with lower computational costs to invert. Moreover, on-the-fly Gaussian elimination (OFGE) was proposed to enable the decoding process using the first arriving coded packets, instead of waiting for the entire generation [62]. This allows reducing the computational complexity of decoding.

2.1.3 RLNC Parameters

The main four contributors that hinder the great performance promised by RLNC are namely,

i) *coding vector overhead*, which is the overhead to signal the coding coefficients,

ii) *linear dependency overhead*, which refers to the excess packets needed due to linear dependency among the coded packets,

iii) *computational complexity* for encoding, recoding, and especially decoding,

iv) *zero-padding overhead*, which is added to original packets to enable the algebraic coding operations.

There exist in the literature different names for the aforementioned, such as the *delay*, which could be considered as the number of extra packets required for decoding, due to losses or the fact that the decoder cannot identify an innovative packet [63]. From a theoretical point of view, we consider the delay to be determined by the probability of decoding the entire transmitted generation. It is also worth mentioning that the problem of computational complexity becomes very daunting when the generation size increases, as the Gaussian elimination becomes impractical. Both the field size and the generation size choices have a heavy impact on the performance and the complexity for encoding and decoding. Consequently, a well-designed code could balance the limitations based on the requirements of each application. As for iv), it depends on the packet sizes of a generation and is the subject of focused research and proposed solutions in Chapters 3 - 5. Therefore, we omit to discuss it in this chapter, and rather focus on the remaining parameters and their aforementioned effects.

2.1.3.1 Field Size Choice

RLNC is designed to perform over finite fields of the form 2^p, where any symbol from such type of fields consists of p concatenated binary bits. Designing an efficient implementation of finite fields remains an active topic. Nevertheless, the network coding library Kodo is considered as the fastest available library that not only proposes a neat implementation of RLNC and some of its variants but also have a fast implementation of the finite fields $\mathbb{F}_q$, where $q = \{1, 4, 8, 16\}$ [64]. It is also recommended to use composite extension finite fields that have the form of a series of smaller fields in order to reduce the overhead and the complexity [65–67]. For example, $\mathbb{F}_{2^2}$, which is a field consisting of 4 elements, is based on polynomial arithmetic over the finite field $\mathbb{F}_2$. Likewise, the finite field $\mathbb{F}_{2^{2^2}}$ with 16 elements uses $\mathbb{F}_{2^2}$ as a base field, and so on and so forth.

In general, high finite fields are better suited for applications that tolerate large delays, or for senders that have limited ability in transmitting extra coded packets. GF(2^8) remains the finite field most employed following these constraints. On the other hand, low finite fields, e.g., GF(2), are required in applications where the encoder, recoder, or/and decoder have limited memory or computational power.

2.1.3.2 Generation Size

The need for performing RLNC over generations formed by original packets reduces the computational complexity and the decoding delay. However, this comes at the cost of introducing additional signalling [68, 69]. This tradeoff could be tuned based on the requirements of an application or a network. Overlapping generations mechanisms were proposed to overcome the delay issue resulting from the computational complexity within a single generation [58, 70–72]. Nevertheless, there exist some approaches that break with this block code feature of RLNC, and propose "generation-less" [73] mechanisms such as sliding window, which will be discussed later in this chapter.

2.1.3.3 Overheads

The exceptional features of RLNC come at the cost of generating different overhead types that could become inevitable for decoding in most cases. The parameters related to RLNC, mainly the generation size and the finite field choice, represent a trade-off in the performance metrics such as the throughput and delay. These are highly dependent on the complexity of decoding as well as the types of overheads that we discuss in the following.

2.1.3.3.1 Coding vector overhead It consists of the apriori knowledge required by the decoder to solve the linear system of equations related to a generation of coded packets. In fact, the N coding coefficients used for the creation of one coded packet are juxtaposed as illustrated in Figure 2.2, forming, therefore, a coding vector. When performing RLNC over GF(2), the coding vector simply signals the original packets employed in forming a coded packet. In general, the coding vector overhead is the length of the coefficients vector as follows:

$$\upsilon_{\text{RLNC}} = N.\log_2(q) \quad (bits). \tag{2.4}$$

This means that as the finite field size or the generation size increases, this overhead becomes significant. Due to their considerable size compared with the packet's payload, the coding vectors could become a burden that typically affects the system's goodput and consumes substantial bandwidth resources. Because IoT-generated packets tend to have small payloads, the coding vector overhead should no longer be neglected, as in some cases this overhead could be larger than the packet to encode itself. This has led to focusing on reducing its size to the size of one symbol while maintaining the ability to recover the original packets using the linear combinations of RLNC using Vandermonde-based coding vectors [74]. Nonetheless, the cyclic property of Vandermonde matrices restricts the generation size to an upper bound equal to $N = \log_2 q$, as well as only tolerating for additional operations in the finite field. This makes the approach more suitable for packets with small payloads, such as WSNs known to have very limited power and generating small payloads. Moreover, some efforts were focusing on compressing the coding vector since it requires more energy to communicate a single bit of data than performing compression in wireless network systems. Gligoroski et al. proposed the Small Set of Allowed Coefficients (SSAC) algorithm that reduces the coding overhead size by two to seven times compared with the related compression techniques [75]. On the other hand, seed-based RLNC, sometimes called Pseudo-Random Network Coding (PRNC) [76], was suggested to shorten the coding vector to the seed size. It simply requires to convey a seed of a Pseudo-Random Number Generator (PRNG) along with the coded packet. The decoder then recomputes the coding vector elements once it receives the coded packets [77,78]. Seed-based network coding does not allow for recoding at the intermediate node, which is a huge obstacle for a large set of scenarios and topologies. This way network coding loses the unique salient feature that promotes it over all other coding techniques. The seed adoption should therefore be limited to multicast or broadcast scenarios.

2.1.3.3.2 Linear dependent packets overhead It consists of the extra number of coded packets transmitted to fully recover a generation. This is highly dependent on the finite field choice, where small fields require a larger number of excess packets compared to the higher fields. Since obtaining a full-rank decoding matrix attests to the successful decoding of a generation, the rank of a random matrix over a finite field shows exactly the impact of the finite field choice on decoding [79]. Receiving enough coded packets for decoding is similar to the coupon collector problem [80, 81].

Furthermore, determining the decoding probability gives insights about a coding scheme's efficiency. We characterizes the additional transmissions needed and could be anticipated by calculating the probability of decoding a generation of size N original after receiving K coded packets as follows [82]:

$$\mathbb{P}(K,N) = \begin{cases} \prod_{i=0}^{N-1}(1 - \frac{1}{q^{K-i}}) & K \geqslant N \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

The proof relies on the counting formula of all $K \times N$ matrices of rank N in GF(q), $\prod_{i=0}^{K-1}(q^N - q^i)$,over the set of all possible $N \times K$ matrices over GF(q), which equals q^{NK} [82, 83].

The expression in Eq. 2.5 does not hold when losses occur in the network, since we should account for the loss rates. For a point-to-point communication with a loss probability ϵ, the decoding probability of decoding N packets over K received coded packets is [82]:

$$\mathbb{P}_\epsilon(K) = \sum_{i=N}^{K} \binom{K}{i} \epsilon^i (1-\epsilon)^{K-i} . \mathbb{P}(i,N). \tag{2.6}$$

Additionally, Claridge et al. provided the exact probability expression of the partial decoding of an RLNC generation and proved that it is very unlikely to recover portions of the generations before acquiring all the coded packets needed [84]. Nevertheless, this proves that RLNC is a robust code against eavesdroppers that would not be able to retrieve even partial information. Other RLNC variants that we will further discuss in this chapter actually promote partial recovery, e.g. systematic RLNC [84].

2.1.3.4 Computational Complexities

They represent an important metric that impacts the RLNC performance, such as latency, energy consumption, and throughput to name a few. The encoding process requires multiplications and additions in the finite field of N packets in order to generate one coded packet. This results in an encoding complexity of the order of $\mathcal{O}(N)$. To the best of our knowledge, the RLNC based approaches have linear encoding and recoding computational complexities in general. As for decoding, it was mainly covered in Section 2.1.2.3. It is of the order of $\mathcal{O}(N^3)$ in the finite field used, as it performs the Gaussian elimination. Some efforts have suggested modifying this algorithm and proposed on-the-fly Gaussian elimination for a lighter decoding computational complexity [62]. We note that it is suggested in general to use small finite fields when the decoder has limited computation power, at the cost of receiving an extra number of coded packets to be able to finish decoding. Likewise, higher finite field sizes could be adopted for systems capable of handling this high decoding computational complexity unlike sensors for example. Eventually, some RLNC based techniques, e.g. fulcrum codes [85], which we describe in the following, offer a balanced combination of different finite fields, depending on a device's capabilities.

2.1.4 RLNC Based Variant Approaches

RLNC has spanned into various techniques that mainly focus on overcoming the limitations of conventional RLNC including, the overhead of delivering the coding coefficients, the complexity of recovering original packets, and the delays of decoding [63].

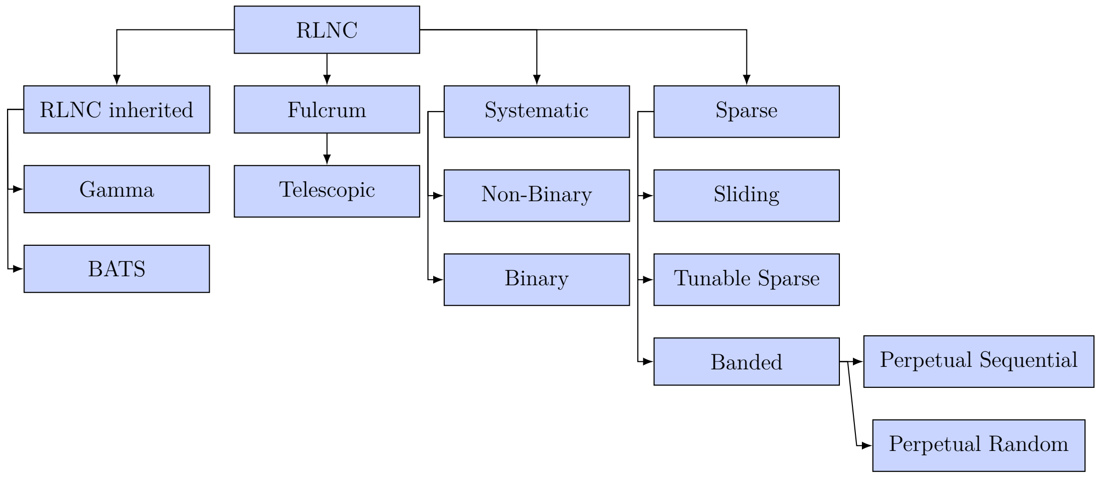


Figure 2.3: The common RLNC-based techniques in the literature.

We provide a classification of the RLNC based approaches found in the literature in Figure 2.3. We believe that there exist four main branches based on common characteristics, namely,

- *Sparse*: schemes that have a high probability of generating zero coefficients.
- *Systematic*: schemes that transmit copies of the packets before switching to sending RLNC coded packets.
- *Fulcrum/composite extension*: schemes that employ composite extension fields instead of the commonly used finite fields.
- *RLNC inherited*: schemes that combine other coding techniques with RLNC.

We discuss these schemes and compare their performance based on our extensive literature study as well as our efforts.

2.1.4.1 Systematic Network Coding

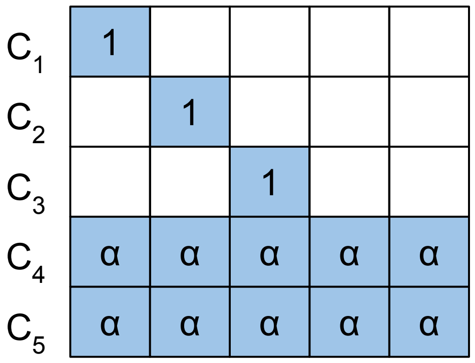


Figure 2.4: Systematic network coding decoding matrix example for a generation of size $N = 5$, for $r = 3$.

This approach was first proposed by Shrader et al. in order to give flexibility to intermediate nodes to replicate and recode packets [86]. The process consists of first r transmitting packets among the K packets in total are identical copies of the original packets. The remaining $K - r$ are generated using standard RLNC. The coding matrix illustrated in Figure 2.4 could be expressed as:

$$\mathbf{M} = \begin{bmatrix} \mathbf{I} \\ \mathbf{J} \end{bmatrix}, \tag{2.7}$$

where **I** is a sub-matrix with the identity matrix properties.

It was shown that this strategy results in low decoding complexity as it requires fewer decoding operations, thus reducing the decoding delays without compromising the throughput [87]. Systematic network coding is well suited for streaming applications as well as multimedia broadcasting [88]. Other research works have considered binary systematic network coding, which refers to using GF(2) coefficients for the matrix **J**. It was implemented on power-constrained devices, such as mobile phones and wireless sensors [89]. Furthermore, it allows partial or progressive decoding for binary sparse network coding [88, 90, 91]. Unfortunately, systematic coding can only be performed at source nodes as it does not support the recoding feature [92]. It can also become impractical to use systematic network coding with high loss rates in the network and should be substituted with standard RLNC.

2.1.4.2 Sparse Network Coding

As previously discussed, small finite fields guarantee lower computational complexity, but they require extra transmissions as they are more susceptible to generating linearly dependent coded packets. To overcome this entangled trade-off, the idea of designing a sparse network code emerged, where sparsity is a key performance metric. This code is characterized by the fact that not all the packets are combined in each transmission. We note that RLNC operates on dense coding vectors because the probability of generating zero coefficient is proportional to the field size, e.g., this probability is equal to $\frac{1}{2^8} = \frac{1}{256}$ for GF(2^8). It should be understood that despite the fact that GF(2) is a sparse code, it generates only either 0 or 1. This makes it challenging to create innovative packets.

Sparse Network Coding (SNC) are similar to the conventional RLNC, with the exception that the coding coefficients are randomly and independently drawn from GF(q) based on the following distribution:

$$\mathbb{P}(\alpha_{ij} = t) = \begin{cases} p, & t = 0 \\ \frac{1-p}{q-1}, & t \in \mathbb{F}_q \setminus \{0\}, \end{cases} \tag{2.8}$$

where $0 \leqslant p \leqslant 1$ represents the level of the code sparsity. When $p = \frac{1}{q}$ for instance, we are back to the RLNC case, whereas SNC is characterized by $p > \frac{1}{q}$. The aim is therefore to design the sparsest code, which increases the throughput but keeps the encoding vector overhead and complexities to a minimum. The benefits of SNC reside mainly in reducing the computational complexities and the coding vector overhead, which in result contribute to saving the energy [93]. A tight bound of the performance of SNC in terms of decoding probability was proposed using absorbing Markov chain [94]. Nevertheless, to the best of our knowledge, there is no exact decoding probability for sparse network coding, or more broadly, there is no probability expression of a sparse matrix in a finite field being full-rank [95, 96].

2.1.4.3 Tunable Sparse Network Coding

Tunable Sparse Network Coding (TSNC) is a sparse code that controls the coding process by tuning the density of the coding vectors over the network [97]. It was proposed by Feizi et al. in order to overcome the limitation of SNC due to the trade-off between the computational complexity and the linear dependency overhead [98]. In a nutshell, TSNC dynamically adapts the level of sparsity depending on the decoding status at the receiver. This approach guarantees to reduce the computational complexity, but at the cost of a slight overhead rise. Based on the theorem in [97], a coded packet with a density equals to $\rho(i, N) \leqslant \frac{1}{2}$, is linearly independent with the previously received i coded packets with a probability $\mathbb{P}^{TSNC}(i, N)$ defined as:

$$\mathbb{P}^{TSNC}(i, N) \geqslant 1 - (1 - \rho(i, N))^{N-i}, \tag{2.9}$$

where $\rho(i, N) = \frac{w}{N}$, and w is the number of non-zero coefficients per coded packet. According to Soerensen et al., TSNC can practically outperform the decoding speed of RLNC by four-folds, without bringing larger overheads [99].

2.1.4.4 Band Codes

It is a structured code that allows RLNC to be performed on a band (window) of input packets in order to control the computational complexity, that results in almost triangular decoding matrices as depicted in Figure 2.5 [100, 101]. Within every band, standard GF(2) RLNC is performed, with a probability of drawing a coding coefficient $\alpha_i = 1$ defined by:

$$\mathbb{P}(\alpha_i = 1) = \begin{cases} \frac{1}{2} & \text{if } f \leqslant i \leqslant d \\ 0 & \text{otherwise} \end{cases}$$

where f is the *leading edge*, and d is the *trailing edge*. Window size defined as $\omega = d - f + 1$, and it is imposed that $0 \leqslant f \leqslant d \leqslant N - 1$ in order to prevent the window from wrapping around, as this may pose an issue for recoding at intermediate nodes.

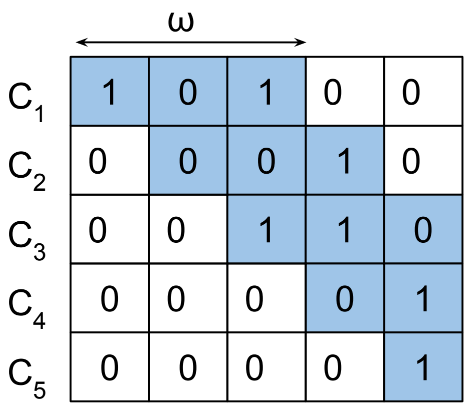


Figure 2.5: Band code decoding matrix example for a generation of size $N = 5$

Nevertheless, generating a low overhead requires the band ω to be relatively wide, similar to the generation size issue for RLNC. The decoding complexity drops with Band codes thanks to the sparsity of the code, and it is in the order of $\mathcal{O}(N\omega)$ [100]. This implies that the energy consumption is reduced, which would extend the battery life of constrained mobile devices.

2.1.4.5 Perpetual Codes

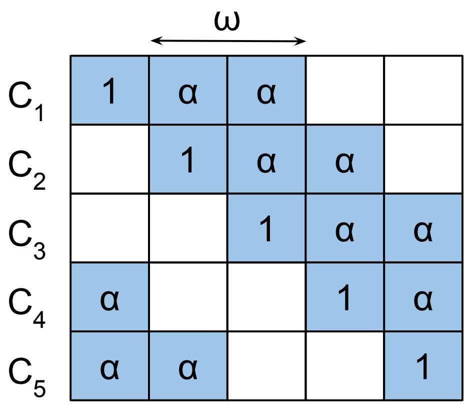


Figure 2.6: Perpetual code decoding matrix example for a generation of size $N = 5$, $\omega = 2$, and α is a random non-zero coefficient from GF(q).

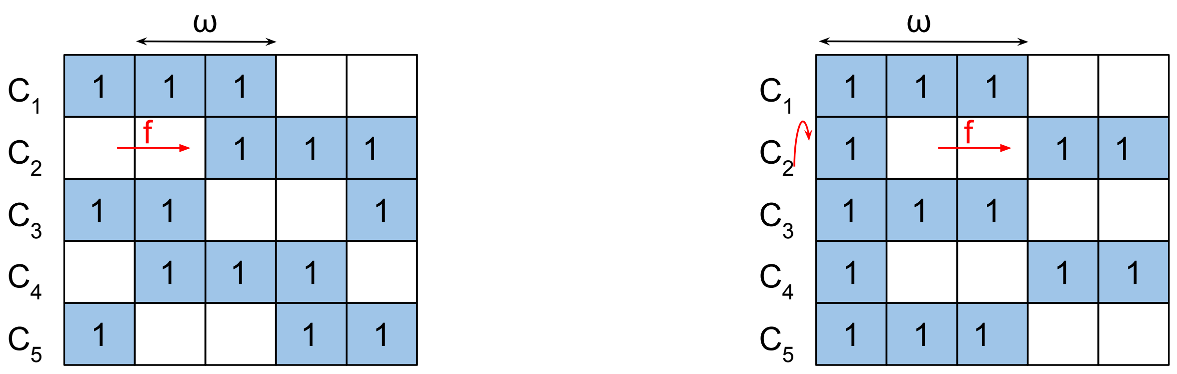
Proposed by Heide et al. [102] where the term was adopted from the work by Maymounkov et al. [103]. It is a band code that is characterized by the wrapping and the pivot element p that equals the 1 coefficient element, and the width ω that refers to the number of non-zero coefficients that are consecutively placed after the pivot p. This is expected to increase the speed of real processors during encoding and decoding.

The pivot element p refers to the position of the 1 elements, i.e. defines which original packet will not undergo a coding coefficient multiplication but will be added to the linear equation corresponding to a coded packet. This sparse but structured code comprises two main modes. *Sequential/pseudo-systematic mode* refers to the mode where p is sequentially drawn from 0 to $N-1$, and with a width ω, $0 < \omega < N$ (see the example in Figure 2.6). When the loss probability is low to moderate, such a mode could be adopted as it reduces the overall overhead as well as the decoding complexity. We note that this mode is reverted to the systematic RLNC when $\omega = 0$. The *Random mode* refers to the case where p is drawn at random from $[\![0\ ;\ N-1]\!]$, and $0 < \omega < N-1$. It is suggested to be used when systematic modes are not beneficial, such as when the loss rate is extremely high. Pahlevani et al. provided an analytical model for the random strategy in the presence of packet loss and showed that ω length is responsible for generating linearly dependent coded packets [104].

In general, the benefits of the code heavily depend on N and ω. According to [102], perpetual codes have simpler decoding algorithms and allow for faster encoding, recoding, and decoding operations thanks to the sparsity of the structured code, which is retained even when recoding is performed. Additionally, RLNC could be extravagantly more computationally demanding than the perpetual codes for a sufficiently large generation size and low width ω size. This obviously results in a higher throughput compared to RLNC. Nonetheless, recoding is not a natural process as in RLNC, and it requires to be judiciously performed.

2.1.4.6 Sequential XORing

We propose this mode of coding as an ultimatum solution for the perpetual codes and band codes. It consists of simply XORing all input packets in a window ω, i.e. only 1 valued coefficients are drawn from GF(2). Based on a pre-defined step size f we create the next coded packet similarly using packets at the positions $f, \cdots, f+\omega$. This policy enables the wrapping as displayed in Figure 2.7. It is worth mentioning that in order to be able to decode the generation, i.e. obtain a full-rank matrix, where $rk(\mathbb{M}) = N$, some conditions must be applied on the window ω or the step size f. As a matter of



(a) Working example

(b) Non-working example

Figure 2.7: Sequential XORing mode decoding matrix example for a generation of size $N = 5$.

fact, matrices designed as such have four possible ranks values $rk(\mathbf{M})$, defined as follows:

$$rk(\mathbf{M}) = \begin{cases} N & \text{if } \omega \nmid N \text{ and } f \nmid N \\ \frac{N}{f \wedge N} & \text{if } \omega \nmid N \text{ and } f \mid N \\ N - (f \wedge N) + 1 & \text{if } \omega \mid N \text{ and } f \nmid N \\ \frac{N-(f \wedge N)}{f \wedge \omega} + 1 & \text{if } \omega \mid N \text{ and } f \mid N \end{cases}$$

As a result, ω and f should be chosen in a way that they do not divide the generation size N. In this case, we only need to signal the window size ω and the step f instead of an entire coding vector of at least N elements of the finite field used. For instance, Figure 2.7 illustrates two examples where f and ω choices, influenced by the generation size N, could result in a non-decodable generation. Sequential XORing is best applied in multicast scenarios, and could be considered as an *index coding* scheme [105–108].

2.1.4.7 Fulcrum Codes

Lucani et al. proposed *Fulcrum codes* to allow a fluid allocation of complexity in the network, while maintaining the high fields performance and a low encoding overhead, thanks to the use of composite extension fields [65, 66]. This is a suitable solution for heterogeneous devices, which definitely have different computational capabilities. It is characterized by a low overhead like when coding using GF(2), which is of the order of $1 + \frac{r}{N} \approx 1$bit per coefficient in a coded packet, where r is the number of excess coded packets transmitted. Additionally, the total number of packets that need to be transmitted is around $N + \mathcal{O}(2^{-r})$, which is similar to the number of transmissions assured by high finite fields. The encoding procedure mainly consists of an *outer* and *inner* code, that gives the opportunity for several decoding approaches. For instance, power-constrained devices perform inner decoding using GF(2) only, whereas powerful devices that could operate on higher finite fields such as GF(2^8) employ outer decoding. Combined decoding balances the performance for medium power devices. Additionally, the processing speed for the encoding process is five to 25 times faster than with GF(2^8), and for decoding it is twice to 20 times faster than GF(2^8) decoding speed.

2.1.4.8 Telescopic Codes

Unlike standard RLNC that allows one finite field only to be used in generating coded packets, we discussed *Fulcrum codes* that allows for a balanced trade-off between the complexity and overhead thanks to the flexibility in the field size choice based on the decoder's capabilities. Telescopic codes expand this idea to a larger set of compatible fields, namely composite extension fields [65], in order to benefit from the low overhead insured by small finite fields and the low linear dependency of coded packets insured by large finite fields. Leveraging several composite extension fields in a single generation of packets enables a better overhead trade-off between the composite extension fields. Additionally, it retains the recoding feature of network coding [109].

2.1.4.9 RLNC Inherited Codes: BATS

This class of codes refers to the approaches that adopt RLNC partially in their design, e.g., BATched Sparse (BATS) [60, 110] and Gamma codes [111]. For instance, these codes usually require an *outer code* and an *inner code*, as well as precoding in some cases. The precoding and the outer coding steps are performed at the source nodes, whereas the inner coding is done on the relay nodes, similarly to recoding with RLNC. For instance, BATS is a well-suited code for wireless multicast and considered as the best joint code based on fountain codes and network coding, which was first advocated by Yang et al. [60, 110]. It consists of an *outer code* and an *inner code.* The latter is a matrix generalization of an LT code, i.e., if the matrix related to the BATS code has the size $N \times 1$, then this would result in an LT code. The outer code is a pre-coding step that has the potential of generating an unlimited number of batches because the rateless feature of the LT code is preserved. The inner code comprises the conventional RLNC operations, which are applied to the symbols that belong to the same batch. It achieves the throughput gain of RLNC and more importantly maintains its recoding

Table 2.1: Computational complexity for encoding/recoding/decoding in finite field operations in the absence of packets losses.

Branch	Encoding	Recoding	Decoding
Standard RLNC	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N^3)$
Slide & XOR	$\mathcal{O}(\omega)$	not supported	$\mathcal{O}(\omega)$
Perpetual	$\mathcal{O}(\omega+1)$	not supported	$\mathcal{O}(\omega^3)$
Band Codes	$\mathcal{O}(\omega)$	$\mathcal{O}(\omega)$	$\mathcal{O}(N\omega)$
Systematic	$\mathcal{O}(N-r)$	not supported	$\mathcal{O}((N-r)^3)$
BATS	$\mathcal{O}(TMd)$	$\mathcal{O}(TMd)$	$\mathcal{O}(nM^3+TM\sum_i d_i)$

feature. BATS codes have a lower computational complexity since they use the Belief Propagation and inactivation decoding algorithms for decoding.

2.1.5 Schemes Performances Comparison

It is worth mentioning that the RLNC based variants were proposed in order to improve RLNC performance. These approaches present certain trade-offs that could be exploited depending on the application and its requirements. In the following, we summarize some of the critical features of these approaches, namely the encoding vector overhead and the computational complexities, since they give a flavor to the throughput, delay, energy consumption, etc.

2.1.5.1 Complexity

The encoding and recoding complexities of RLNC do not pose as big of a problem as its cubic decoding complexity, wchich is the reason behind proposing the RLNC variants. Table 2.1 summarizes the overall complexities related to some of these approaches, that basically present reduced computational complexities, despite the omnipresent cubic factor of the Gaussian elimination. Nevertheless, the issue with recoding, in general, relies on the fact that the intermediate node that is supposed to perform recoding should not be aware of the packet design it received in the first place, otherwise it would be counted as decoding. As recoding should be performed blindly, the node cannot decide whether a certain coded packet is for example a systematic copy of an original packet or densely coded packet using RLNC. The losses and erasures that might occur during coded packets transmissions do not allow nodes to keep track of the general information regarding the packets it receives. This explains why some RLNC variants cannot have the recoding feature.

Table 2.2: Total encoding vector overhead created for the transmission of exactly N coded packets equal to the generation size per category.

Branch	Coding Overhead (bits)
Standard RLNC	$N\log_2(q)$
Traditional sparse	$s\log_2(q)$
Sliding window	$\omega\log_2(q)$
Slide & XOR	$\lceil\log_2(f)\rceil + \lceil\log_2(\omega)\rceil$
Perpetual	$\log_2(N) + \omega\log_2(q)$
Fulcrum	$N(1+\frac{r}{N})$
Telescopic	$\sum_{j=1}^{N}\lceil\log_2(q_i)\rceil$
Band Codes	$N+C$
Systematic	N

2.1.5.2 Overhead

Aside from the basic techniques to reduce the encoding overhead of RLNC, such as the header compression and the randomly generated seeds, the variants of RLNC propose interesting solutions in reducing the encoding vector overhead as well. Table 2.2 shows the various possibilities for reduced overhead based on the schemes' designs. For instance, the systematic approach generates minimal overall overhead per generation transmission, especially when the number of systematic packet transmissions is relatively high. We note that it is interesting to have encoding vector overheads that are independent of the finite field size choice, as this could significantly contribute to increasing the overhead's size.

2.2 Compressed Sensing

Compressed sensing is a disruptive transform coding technique that enables efficient signal acquisition and later an effective exact or approximate reconstruction, far below the Shannon-Nyquist sampling rate. It is the outcome of a long research [26,27,112,113]. Namely, it was first introduced in 2005 by Candes et al. [28], and later Donoho et al. gave it its current name [113]. Compressed sensing fulfills the sufficient conditions for a perfect or approximate reconstruction of a signal using very few measurements, i.e., sample. This promising performance gave compressed sensing a tremendous interest from various fields, aside from signal processing, where it first arose. For instance, it is currently used by power and bandwidth-constrained systems to optimize the measurement process. Nevertheless, its core issues are mainly the proper design of the measurement matrix that will not damage the signal during the compression step and the design of an efficient reconstruction algorithm that will accurately restore the original signal. The key properties to be satisfied are incoherence and sparsity. Our goal in this section is to provide answers to the following questions:

i) Under which conditions an under-determined linear system of equations could be considered as compressed sensing, i.e. admits an exact or an approximate solution?

ii) Which reconstruction algorithms are able to effectively retrieve original signals?

Furthermore, there is a vast number of papers and books surveyed the compressed sensing [2, 114]

2.2.1 Overview

From a mathematical point of view, compressed sensing proposes to find the sparsest solutions to under-determined linear systems of equations, of the form of $\mathbf{y} = \mathbf{A}\mathbf{x}$, by the means of optimization techniques. The compressed sensing problem is summarized as follows:

$$\min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{y} = \mathbf{A}\mathbf{x}, \tag{2.10}$$

where $\mathbf{x} \in \mathbb{R}^n$ is a sparse signal to compress, $\mathbf{y} \in \mathbb{R}^m$ is the measurement vector, i.e. compressed version of $\mathbf{x}$, and $\mathbf{A} \in \mathbb{R}^{m \times n}$ is the measurement matrix, which transforms $\mathbf{x}$ into $\mathbf{y}$. We remind that the ℓ_0 pseudo-norm counts the number of non-zero elements in a vector. The exact locations of such elements lie in the *support* set, denoted *supp*. This non-convex optimization problem becomes an enormous burden for large values of n. In general, the Euclidean vector space $\mathbb{R}^n$ is provided with ℓ_p norms as follows:

$$\|\mathbf{x}\|_p := \begin{cases} (\sum\limits_{i=1}^{n} |x_i|^p)^{1/p}, & p \in]0, \infty[\\ \max\limits_{i \in \{1, \cdots, n\}} |x_i|, & p = \infty. \end{cases} \tag{2.11}$$

Furthermore, we started talking about compressed sensing when Candes et el. proved that substituting the ℓ_0 pseudo-norm with the ℓ_1 norm allows for exact reconstruction with a reasonable computational complexity, changing the under-determined system of equations in Eq. 2.10 to a convex problem. They proposed the use of Basis Pursuit (BP), which is a linear program based on the ℓ_1 and demonstrated that it could approximate

ℓ_0 under certain conditions that we clarify later in this section. The new problem in question is therefore:

$$\min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{subject to} \quad \mathbf{y} = \mathbf{A}\mathbf{x}. \tag{2.12}$$

2.2.2 Sparsity and Compressibility

A signal $\mathbf{x} \in \mathbb{R}^n$ is called *sparse* if most of its components are zeros. This means that it has a low information rate. Therefore, most of the signal's elements can be discarded without losing important information. Specifically k-sparse if it has $k \in [\![1 \ ; \ n-1]\!]$ non zero elements, defined as:

$$\|\mathbf{x}\|_0 := Card(supp(\mathbf{x})) = Card(\{i \in [\![1 \ ; \ n]\!] \ : \ x_i \neq 0\}) = k. \tag{2.13}$$

It constitutes a crucial condition to ensure the reconstruction of $\mathbf{x}$. However, finding an exact sparse representation could be challenging for some real-life signals. Thus, the sparsity concept could be a tough constraint in some cases. The substitution with the weaker concept of compressibility actually does not affect the compressed sensing reconstruction, but it widens its usage for far more cases. A compressible signal is not exactly sparse, but has a small error of the best k-term approximations $\sigma_k(x)_p$, that we define as:

$$\sigma_k(x)_p := \inf\{\|\mathbf{x} - \mathbf{y}\|_p, \ \|\mathbf{x}\|_0 = k\}, \tag{2.14}$$

with $p \geqslant 1$. It is worth noting that the difficulty in solving Eq. 2.10 lies in finding the exact locations of the non-zero coefficients of a sparse vector, i.e. the support.

2.2.3 Sensing Matrix Design

A momentous part of the compressed sensing research was devoted to the design of the appropriate sensing matrix, in order to guarantee the success of the problem in Eq. 2.12. This means that the gap between the estimated recovered signal $\hat{\mathbf{x}}$ and the original one to be robust and stable, has a negligible value. The reconstruction accuracy highly depends on the incoherence property of the matrices. In the following, we discuss some of the common conditions to impose on the sensing matrix $\mathbf{A}$ in order to guarantee a stable recovery.

Incoherence is denoted by $\mu(\mathbf{A})$, and defined as the largest absolute and normalized dot product between two different atoms/columns of $\mathbf{A}$ as follows:

$$\mu(\mathbf{A}) = \max_{i \neq j} \frac{\|\langle \mathbf{A_i}, \mathbf{A_j} \rangle\|}{\|\mathbf{A_i}\|_2 \|\mathbf{A_j}\|_2}, \tag{2.15}$$

where $\mathbf{A_i}$ is a column of $\mathbf{A}$, $i = \{1, \cdots, j, \cdots, n\}$. For full rank matrices specifically, $\mu(\mathbf{A})$ satisfies [115]

$$\sqrt{\frac{n-m}{m(n-1)}} \leqslant \mu(\mathbf{A}) \leqslant 1. \tag{2.16}$$

A low mutual coherence guarantees to elevate the reconstruction accuracy [116], and it is a more practical option for deterministic matrices, e.g., Magnetic Resonance Imaging (MRI) matrices [117].

Null Space Property "was first proposed in [118]. It depends exclusively on the null space (kernel) of the measurement process. It is known to be the necessary and sufficient conditions on the successful reconstruction of sparse signals when using the basis pursuit algorithm, or other ℓ_1-relaxations, in the absence of noise.

We denote by $T \subset \{1, \cdots, n\}$ and $T^c = \{1, \cdots, n\} \backslash T$ as its complement. Additionally, we define $\upsilon \in \mathbb{R}^n$ and $\upsilon_\mathbf{T} \in \mathbb{R}^{|T|}$ as the vector containing the coordinates of υ on T, similarly for $\upsilon_{\mathbf{T^c}}$. We denote by $\mathbf{A_T}$ the $m \times |T|$ sub-matrix of $\mathbf{A} \in \mathbb{R}^{m \times n}$, that encloses the columns of $\mathbf{A}$ indexed by T. If we define $T = supp(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^n$, then $\mathbf{Ax} = \mathbf{A_T x_T}$ [2]".

Definition 1. *(Definition)*
A matrix $\mathbf{A}$ has the NSP of order k if,

$$\|\upsilon_\mathbf{T}\|_1 < \|\upsilon_{\mathbf{T^c}}\|_1,\ \forall \upsilon \in Ker(\mathbf{A}) \setminus \{0\},\ T \in \{1, \cdots, n\}\ \textit{with}\ Card(T) \leqslant k.$$

Restricted Isometry Property is an alternative property that still captures the idea of incoherence, and provides a finer measure about the suitability of the measurement matrix, which was first introduced by Candes et al. in 2005 [28, 119]. It basically characterizes nearly orthonormal non-square matrices in the context of sparse or compressible signals.

Lemma 1. *If there exits a constant $\delta_k \in (0,1)$, called the k-th restricted isometry constant of a matrix $A \in \mathbb{R}^{m \times n}$, such that for any k-sparse signal $x \in \mathbb{R}^n$, we have*

$$(1 - \delta_k)\|\mathbf{x}\|_2^2 \leqslant \|\mathbf{Ax}\|_2^2 \leqslant (1 + \delta_k)\|\mathbf{x}\|_2^2, \tag{2.17}$$

then $\mathbf{A}$ *is said to satisfy the RIP of order* k*.*

These properties to fulfill are considered adequate conditions to accurately reconstruct a signal. An efficient design of the measurement matrices is a crucial aspect of the compressed sensing theory. They influence the precision of the reconstruction process at the decoder, determining an exact or approximate reconstructed signal. Different structures of matrices are used, e.g., random, deterministic, and structured, with a huge success given to random matrices in theory. However, when it comes to hardware realization, light-weight and easy implementation even with a slight performance compromise is extremely important. We note the gap between the Restricted Isometry Property (RIP) and Null Space Property (NSP) is discussed in [120]. Despite the fact that both properties are NP-hard, RIP remains easier to justify theoretically and has a wider set of applications to various reconstruction algorithms.

2.2.3.1 Suitable Sensing Matrices

In general, generating adequate measurement matrices that enable exact or a high reconstruction accuracy remains unfortunately an intriguing endeavor, and remains an open research problem in the field of compressed sensing. Surprisingly, random matrices are capable of fulfilling the RIP constraint, with a special focus given to Bernoulli, Rademacher, Gaussian, and sub-Gaussian matrices. For instance, the Gaussian matrices are most used in the literature since it goes hand in hand with the RIP and allows an accurate reconstruction of Eq. 2.12 with a sufficiently high property [121]. Bernoulli matrices have a discrete probability distribution, which simplifies the coefficients management and storage. The simple implementation reduces the complexity costs, since the signals in question are either added or subtracted, avoiding the heavy costs for multiplication and storage. Moreover, Weibull random matrices guarantee an accurate signal reconstruction with high probability using an optimal number of measurements [122].

There exists also a class of deterministic matrices that are not commonly adopted because, as previously explained, it is computationally complex to verify if a matrix fulfills the RIP property. Nevertheless, they facilitate the hardware implementation and could be used in applications where precise signal recovery is not mandatory. For instance, Vandermond matrices of size $k \times n$ have guarantees for recovering a k-sparse signal. Other examples include chirped sensing matrices [123], second-order Reed-Muller, and LDPC matrices. However, these matrices cannot achieve the expected bound of random matrices because of the weaker RIP that they can fulfill.

2.2.4 Common Reconstruction Algorithms

In compressed sensing theory, the computational load is shifted to the reconstruction side instead of the sampling process. This makes the task of having an efficient and less costly reconstruction algorithm competitive and challenging. This is explained by the large indescribable number of algorithms found in the literature. Zhang et al. provided a detailed survey covering major important algorithms in [124]. In the following, we discuss some of the reconstruction algorithms classes that we are adopting in the dissertation.

2.2.4.1 Convex relaxation

Convex relaxation is a class of algorithms that relies on convex optimization, such as linear programming. Amongst these algorithms we refer to BP [125], Gradient Projection for Sparse Reconstruction (GSPR) [126], Least Absolute Shrinkage Operator (LASSO) [127], Basis Pursuit De-Noising (BPDN) [128], etc. For instance, *Basis Pursuit* is a decomposition method that, due to the advances in linear programming, is able to choose the right solution – with the minimum ℓ_1 norm coefficients – for under-determined linear systems of equations among the many possible solutions. Nevertheless, these algorithms remain computationally complex, and cannot be considered optimal for large-signal sizes [129].

2.2.4.2 Greedy Pursuit

The basic goal of greedy algorithms is to gradually find the support of the unknown vector. For instance, at each iteration, one or more coordinates of the vector are selected for testing, based on the correlations between the columns of $\mathbf{A}$ and the regularized measurement vector. OMP [130], Regularized Orthogonal Matching Pursuit (ROMP), which is a variant of the OMP [131], Compressive Sampling Matching Pursuit (CoSaMP) [132], and Subspace Pursuit (SP) [133], are the predominantly utilized greedy algorithms.For instance, OMP is often the chosen algorithm for compressed sensing and sparse coding in general thanks to its high performance.

2.2.4.3 Thresholding

Iterative thresholding algorithms were introduced for compressed sensing as an alternative to convex optimization. They are divided into two sub-classes, namely *soft-thresholding* also known as shrinkage, e.g., Iterative Soft Thresholding (IST) [134], and hard thresholding, e.g., Iterative Hard Thresholding (IHT) [135]. This class of algorithms start by posing the first condition $\mathbf{x}^0 = 0$, and continue with the following iteration, for $n = 1, 2, \cdots$:

$$\mathbf{x}^{n+1} = \mathbf{H}_k(\mathbf{x}^n + \mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x}^n)), \tag{2.18}$$

where $\mathbf{H}_k$ is a non-linear operator that sets the input vector coefficients other than the k largest ones to zero.

2.2.4.4 Message Passing

Message passing is a class based on Fast Iterative Thresholding (FIT), and is also inspired by the *Belief Propagation* for graph decoding. This is known as *sum-product message passing* [129], as well. Amongst this category of algorithms, we refer to Approximate Message Passing (AMP) [129], Expectation Maximization Belief Propagation (EMBP) [136] and Smoothed l_0 (SL0) [137].

2.2.5 Computational Complexity

As for the computation complexities of compressed sensing reconstruction algorithms, they are known to be costly since they have to find the support of a vector using various optimization mechanisms. This can actually be a handicap for next-generation real-time applications, where millisecond response is expected. Nonetheless, some of these state-of-the-art algorithms provide lower complexities. For instance, the *BP* algorithm has a cubic complexity due to the fact that the ℓ_1 norm is also computationally expensive. On the other hand, greedy iterative algorithms seem to have the lowest complexities, as expected. Note that although the reconstruction algorithms have high computational complexities, they are not employed at the sensors, but rather at the sink side, which would normally have more available computational resources or could even potentially leverage the capabilities of future Mobile Edge Cloud (MEC) [138] implementations.

2.2.6 Sparse Representation

Sparse representation is the theory that is focused on finding a representation of a vector (e.g. a signal or an image) as a linear combination of a few elementary vectors, called *atoms* in the literature, that are obtained either from a well-known transform or using a learned dictionary. It is useful in many fields including pattern recognition, signal processing, image classification, and segmentation [124]. Many modern fields rely on sparsity thanks to its effectiveness in alleviating common research problems. For example, compressed Sensing assures that if a signal $\mathbf{x} \in \mathbb{R}^n$ can have a sparse representation in an orthogonal basis $\mathbf{\Psi} \in \mathbb{R}^{n \times n}$ using $\theta \in \mathbb{R}^n$, then, only a few non-adaptive measurements $\mathbf{y} \in \mathbb{R}^m$ are needed in order to reconstruct the signal. Based on the nature and the structure of a certain signal, a set of representation systems could provide a sparse approximation. In general, they expand based on the latest research trends. For instance, it was recently shown that the novel shearlets transform, which is a natural extension of wavelets, provides an optimal sparse representation of most natural images unlike the conventional wisdom regarding the wavelet transform [139]. In the general case where $\mathbf{x}$ is not necessarily sparse, the signal $\mathbf{x}$ could be then expressed as:

$$\mathbf{x} = \mathbf{\Psi}\theta, \tag{2.19}$$

where,$\mathbf{\Psi} \in \mathbb{R}^{n \times n}$ is the sparsifying basis, and $\theta \in \mathbb{R}^n$ is the sparse representation of $\mathbf{x}$ resulting from its projection in the basis $\mathbf{\Psi}$. For simplicity, we consider in the following that the sensing matrix $\mathbf{A}$ is represented as $\mathbf{A} = \mathbf{\Psi\Phi}$, where $\mathbf{\Phi}$ is the measurement matrix. Thus, the optimization problem is rewritten as:

$$\min \|\theta\|_1 \quad \text{subject to} \quad \mathbf{y} = \mathbf{\Psi\Phi}\theta. \tag{2.20}$$

2.2.6.1 Well-known Transformations

"Understanding the transformation from one basis to another, which has a lower dimension, could be simply done by considering a common vector of three dimensions (localization). However, in reality, it is lying in a very big dimension different from its usual three-dimensional coordinates. This means that sparse signals contain much less information than their ambient dimension suggests. [140] explains how to exploit the sparsity properties of signals in order to process wireless signals in different applications. One of the most known examples is the JPEG2000 coding standard, which used the sparsity of the wavelet coefficients of natural images [141], and the Joint Photographic Experts Group (JPEG) standard, which uses the DCT basis. There exists a large set of sparsifying transforms, including, but not limited to, steerable wavelets, Gabor dictionaries, chirplets, warplets, multi-scale Gabor dictionaries, wavelets packets, cosine packets, etc. Despite the large number of sparsifying transforms, some data unfortunately cannot be sparsely approximated using the aforementioned common bases. The sparse representation of signals based on orthogonal bases in general highly depends on whether the signal's characteristics could be matched with the specific basis' function. [2]".

2.2.6.2 Over-Complete Dictionary Learning

Learning over-complete dictionaries is a new approach closely related to compressed sensing that allows sparsifying the signals that cannot undergo this procedure using the previously discussed well-known transforms. It is also used in the field of machine learning for the purpose of classification, which aims for example to designate the correct label to a certain image. Finding an over-complete dictionary $\mathbf{D}$ requires a large set of training samples, denoted $\mathbf{X} \in \mathbb{R}^{n \times l}$, $l \leqslant n$. The sparsest representation is obtained using either of the following optimization problems:

$$\min_{\mathbf{X}} \|\mathbf{X}\|_0 \quad \text{subject to} \quad \mathbf{Y} = \mathbf{D}\mathbf{X}, \tag{2.21}$$

or

$$\min_{\mathbf{X}} \|\mathbf{X}\|_0 \quad \text{subject to} \quad \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_2 \leq \epsilon, \tag{2.22}$$

where $\mathbf{Y} \in \mathbb{R}^{n \times l}$, and ϵ has a negligible value in $\mathbb{R}$.

There exist a set of standard algorithms for training over-complete dictionaries, e.g., Method of Optimal Directions (MOD) [142], union of orthonormal bases, maximum a posteriori probability approach, and K-SVD [29]. The latter algorithm relies on the K-means clustering, as well as the Singular Value Decomposition (SVD) algorithms. It basically iteratively operates over two-stages, alternating between finding a sparse approximation of $\mathbf{X}$ based on the current state of the dictionary, and update the dictionary atoms accordingly. The related optimization problem is defined as follows:

$$\min_{\mathbf{D},\mathbf{X}} \{\|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_{\mathcal{F}}^2\} \quad s.t. \quad \forall i, \ \|\mathbf{x_i}\|_0 \leqslant T_0, \tag{2.23}$$

where $\|.\|_{\mathcal{F}}$ is the Frobenius matrix norm, defined as $\|\mathbf{A}\|_{\mathcal{F}} := \sqrt{tr(\mathbf{A}^*\mathbf{A})}$.

In practice, the *K-SVD* algorithm could be easily implemented using the scikit-Learn framework [143]. Appendix B shows the impact in practise of the reconstruction that trained over-complete dictionaries guarantee compared to well-known transformations, given a large set of data to train the dictionary [11].

Table 2.3: Summary of the main conditions to be fulfilled to solve a compressed sensing problem of the form $\mathbf{y} = \mathbf{A}\mathbf{x}$.

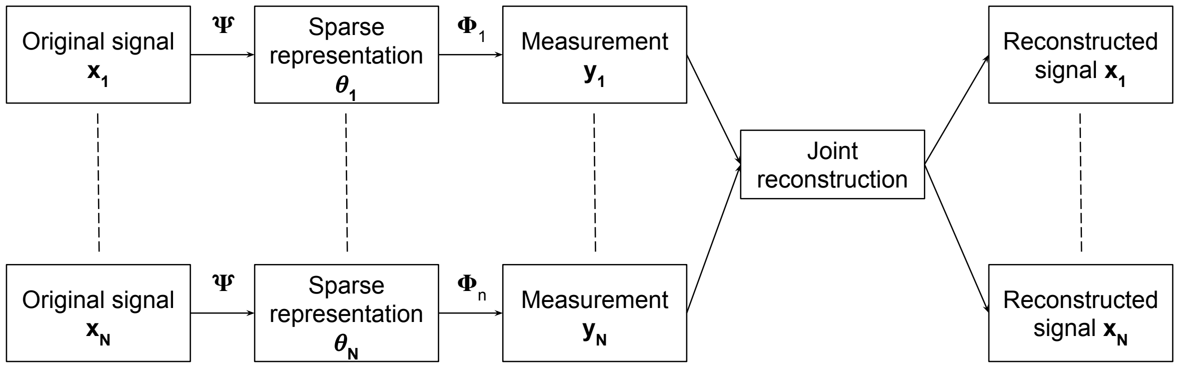
Element	Conditions
$\mathbf{x}$	sparse *Not sparse?* Find a sparse representation in a different transform *No pre-existent transform?* Train an over-complete dictionary *Not exactly sparse?* Compressible signals work as well
$\mathbf{A}$	Obeys the mutual coherence or one of the common properties e.g. RIP, NSP *NP-hard problems?* Use Gaussian or Bernoulli matrices
m	$m << n$: the number of measurements m is far fewer than the signal's dimension n
k	$k < m$: the sparsity of the signal is lower

2.2.7 Take Away Conditions

Compressed sensing theory asserts that sparse signals could be recovered from a small number of linear measurements using ℓ_1-minimization and other optimization techniques. It is mandatory to consider the main conditions to be fulfilled when tackling under-determined linear systems of equations. In order to determine whether one deals with a compressed sensing problem, it is crucial to verify that the vector $\mathbf{x}$ is sparse and that the matrix $\mathbf{A}$ fulfills the coherence property. However, as previously explained, sparsity is no longer an issue for vectors that are sufficiently large. As a matter of fact, a wide class of signals could be sparsified using standard transformations or bases, or simply by training an over-complete dictionary. Moreover, compressible signals are also a good fit. This makes any large signal susceptible to being compressed, which widens the applications of compressed sensing and especially makes it a powerful tool against big data. On the other hand, it is easy to obtain a measurement matrix, despite the expensive computational cost of verifying the right properties to fulfill. It has been proved that Gaussian and Bernoulli matrices for example obey the RIP with a very high probability. Table 2.3 summarizes the main conditions to be verified in order to guarantee to have a compressed sensing problem, i.e., being able to exactly or approximately reconstruct the original $\mathbf{x}$ using the right reconstruction algorithm.

2.2.8 Distributed Compressed Sensing

DCS is the field resulting in combining compressed sensing and Distributed Source Coding (DSC) [144]. Such a combination allows the exploitation of inter-signal and intra-signal



Original signal $\mathbf{x_1}$
Ψ
Sparse representation θ_1
Φ_1
Measurement $\mathbf{y_1}$
Joint reconstruction
Reconstructed signal $\mathbf{x_1}$
Original signal $\mathbf{x_N}$
Ψ
Sparse representation θ_N
Φ_n
Measurement $\mathbf{y_N}$
Reconstructed signal $\mathbf{x_N}$

Figure 2.8: DCS framework for a set of N signals. For all $i \in \{1, \cdots, N\}$, each original signal $\mathbf{x_i}$ is transformed using the common sparsification matrix $\mathbf{\Psi}$ into a sparse signal $\theta_\mathbf{i}$. The resulting sparse signals are each compressed using a specific matrix $\mathbf{\Phi_i}$. At the receiver, a joint reconstruction is possible thanks to the common features among the signals. Finally, every signal can be exactly reconstructed [2].

correlations efficiently. As a matter of fact, large sets of similar data types have generally a structure or a pattern that could be modeled, enabling intelligent ways of processing and representing the data. DCS promoted compressed sensing to be employed in a wider set of applications, aside from being a signal processing tool. For instance, DCS is now strongly involved in the communications fields, where WSNs are a major hardware element of IoT devices. Signals generated by WSN are perfect for DCS application. Figure 2.8 depicts the basic idea behind the DCS framework, where n original signals are in play. Each original signal $\mathbf{x_i}$ is independently sampled into $\mathbf{y_i}$, and then transmitted to a common sink, where all the measurements are used to jointly reconstruct in one-step an estimation of the set of original signals. This allows for a considerable reduction in computational complexity, which leads to reduced delays at the sink for reconstruction.

In the scenario of DCS, we suppose that there is a set of signals $\mathbf{X} = [\mathbf{x_1 x_2} \cdots \mathbf{x_N}]^T$ from N nodes, where $\mathbf{x_i} \in \mathbb{R}^n$, $i \in \{1, \cdots, n\}$, is a sparse signal. The compressed measurements $\mathbf{Y}$ are expressed as:

$$\mathbf{Y} = \mathbf{\Phi X}, \tag{2.24}$$

where $\mathbf{Y} = [\mathbf{y_1 y_2} \cdots \mathbf{y_N}]^T$, and the measurements matrix $\mathbf{\Phi}$ is defined as

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{\Phi_1} & 0 & \cdots & 0 \\ 0 & \mathbf{\Phi_2} & 0 & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & \mathbf{\Phi_N} \end{bmatrix}. \tag{2.25}$$

This means that every measurement vector can be expressed as $\mathbf{y_i} = \mathbf{\Phi_i x_i}$, $i \in \{1, \cdots, N\}$, for $\mathbf{y_i} \in \mathbb{R}^m$ and $\mathbf{\Phi_i} \in \mathbb{R}^{m \times n}$.

2.2.8.1 Joint Sparsity Models

The signal in question $\mathbf{x_i}$ is represented by a *common component* that is present in all of the signals in a set, and an *innovation component*, that is unique to each signal. As such, $\mathbf{x_i}$ can be expressed as

$$\mathbf{x_i} = \mathbf{z_i} + \mathbf{z_c}, \quad i = 1, 2, \cdots. \tag{2.26}$$

JSM-1 known as *sparse common component* + *innovation* model. It is characterized by the fact that

$$\mathbf{x_i} = \mathbf{z_c} + \mathbf{z_i}, \quad i \in [\![1 \ ; \ S]\!],$$

with,

$$\begin{aligned} \mathbf{z_c} &= \mathbf{\Psi}_c \cdot \theta_c \quad \|\theta_c\|_0 = k_c \\ \mathbf{z_i} &= \mathbf{\Psi}_i \cdot \theta_i \quad \|\theta_i\|_0 = k_i \end{aligned}$$

where k_c and k_i are the respective sparsity values. This example describes best a network of sensors, which are geographically located close to one another and simultaneously monitoring a natural phenomenon that smoothly fluctuates in space and time, such as humidity and temperature. The readings share inter and intra-signal correlations, translated as temporal and spatial correlations.

JSM-2 known as *Common sparse support* model. It is characterized by signals that can be formed using the same sparse basis, however, with different coefficient for each signal.

$$\mathbf{x_i} = \mathbf{\Psi}\theta_i, \quad i \in [\![1\ ;\ S]\!].$$

It is most used for multi-lead ElectroCardioGram (ECG) [145], and Multiple Input Multiple Output (MIMO) [146] for example.

JSM-3 Known as *Non-sparse common component + sparse innovations* model. Each signal is supposedly composed of an arbitrary common component, denoted $\mathbf{z_c}$, and a sparse innovation component $\mathbf{z_i}$. Unlike the JSM-1, the common component does not have to be sparse. This model can be used in situations where it is difficult or impossible to acquire a sparse representation $\mathbf{z_c}$ in any basis or frame. This makes it adequate for scenarios where the inter-signal correlations are predominant compared with the non-existence of intra-signal correlations.

2.2.8.2 DCS Reconstruction Algorithms

"The joint recovery could be performed via ℓ_0-minimization, but as discussed earlier in this chapter, it is more reasonable to relax it and recover the signal ensemble via the ℓ_1-minimization. Without loss of validity of the standard compressed sensing reconstruction methods, many algorithms were proposed for DCS, just as for Compressed Sensing itself. Their properties differ based on the nature of the signal set, application, and requirements of the system – mainly the computational complexity and the tolerated reconstruction error. Additionally, various reconstruction strategies were proposed for each JSM model, in order to meet the desired specifications. For example, One-Step Greedy Algorithm (OSGA) solves the DCS data ensemble modeled using the JSM-1. Moreover, Trivial Pursuit (TP) is a greedy algorithm that was designed for the JSM-2. It demands a large signal set in order to perform well. Furthermore, DCS-Simultaneous Orthogonal Matching Pursuit(DCS-SOMP) [147] is a DCS-based Smoothed Orthogonal Matching Pursuit (SOMP) (which is a variant of OMP). This strategy requires a small number of measurements that are proportional to the sparsity k for a moderate number of signals. Baron et al. [148] showed that reconstructing one signal of the set could be achieved using $k + 1$ measurements as the number of signals tends to infinity. More algorithms are available in the literature, but the choice of discussing TP and DCS-SOMP was made as they were some of the first reconstruction strategies for DCS [2]".

2.2.9 Compressed Sensing and Blind Source Separation

We target at outlining key innovation technologies that can enrich and improve the network and processing layers in the data flow view of the proposed architecture for Industry 4.0 applications with a huge number of Industrial Internet of Things (IIoT) sensor devices, which allow for continuous monitoring and human-less decision making in a wide range of scenarios in conjunction with Industry 4.0 [32, 34]. For instance, it has become crucial to monitor IIoT devices on a factory floor in order to remotely detect any anomalies or deficiencies in the responsible device. Here, deployed sensors continuously transmit, for example, acoustic data to remote servers, which in turn are capable of discovering any occurring deviation. This allows for early-stage problem detection to prevent production breakdown. Nevertheless, with the massive number of sensors that generate massive data, it has become a challenging task to quickly and accurately locate the malfunctioning device. Also, the high latency during the remote data separation could yield a poor QoE.

Such a situation could be dealt with using the Blind Source Separation (BSS) technique that enables separate signal extractions from a set of mixed signals [149], where both the source and the mixing process are unknown. It separates multivariate data from observers, $\mathbf{X} \in \mathbb{R}^{n \times m}$, into additive n components, $\mathbf{S} \in \mathbb{R}^{n \times m}$ [149]. The n additive components $\mathbf{S}$ are original sources in m time slots. These n components of sources should be statistically independent and non-Gaussian distributed. The multivariate data $\mathbf{X}$ are the mixtures obtained from n observers in the same m time slots. The component $\mathbf{x}_i$ of the observed mixtures $\mathbf{X}$ at each time instant is generated as a linear combination of the source components $\mathbf{s}_i$. The mixing model can be understood as a function $\mathcal{M}$ defined as:

$$\mathbf{X} = \mathcal{M}(\mathbf{A}, \mathbf{S}) = \mathbf{AS}, \tag{2.27}$$

where $\mathbf{A}$ is the mixing matrix.

Along with compressed sensing, both techniques are cheap and trustworthy for sensors thanks to their minimal encoding requirements as the main calculations for data reconstruction are performed at the receiver, which is expected to have powerful computation resources. Their adoption in scenarios, like continuous monitoring of factories, is of tremendous interest as they relax the problem of Big Data. To overcome the major bottlenecks of low-latency and massive data, we adopt in the following CdICA [150] and Compressible Source Separation (CSS) [33].

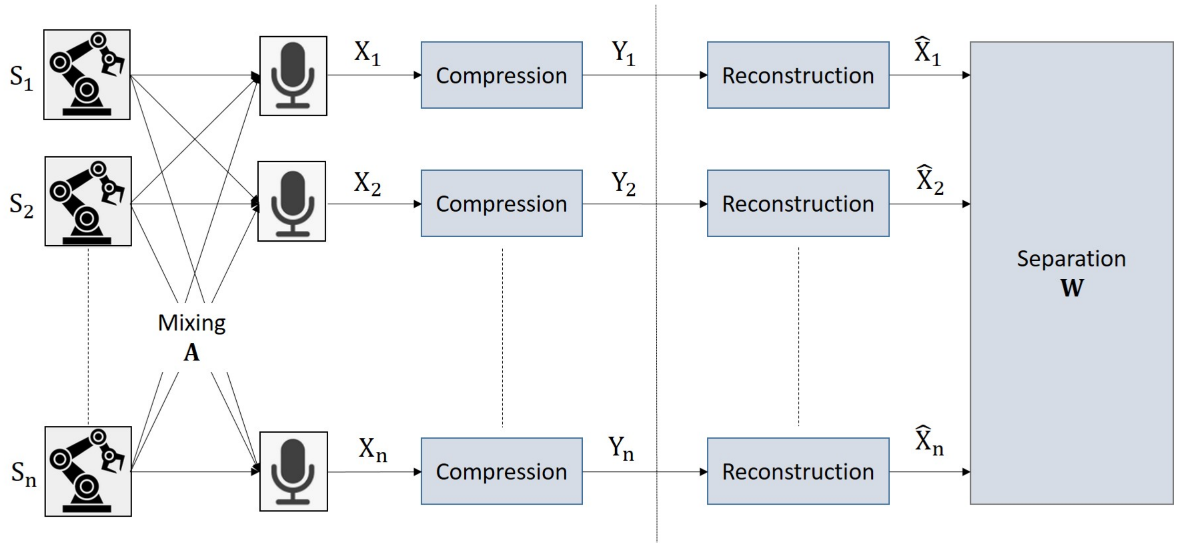


Figure 2.9: Compressible Source Separation example with n sources.

2.2.9.1 Fast Blind Source Separation

CdICA is a new framework proposed to separate multiple randomly mixed signals into independent source signals for fast decision making in time-sensitive applications. Based on the notable FastICA algorithm [149], CdICA generates an initial separation matrix relying on the known mixture components in order to increase the separation speed of the traditional Independent Component Analysis (ICA).

Up to 30 industry-related audio signals were randomly selected from Google Audio Data Set (GADS) [151], which is a widely used large-scale data set including engines, tools, motors, and so on, as the original sources **S**. This enabled our test data to cover as many types of audio data in different industrial production scenarios as possible. Each of them was a 10-second audio signal with 32 kHz sampling rate. The length of each source was $m = 320000$. The mixtures **X** were generated using eq. (2.27), as the known input of the CdICA and FastICA. For each given source number n, 30 samples were tested. The metric R_t indicates the percentage of the separation time reduction rate, which is the ratio of the time by CdICA and the time by FastICA.

$$R_t = \frac{T_{CdICA}}{T_{FastICA}} \times 100\%. \tag{2.28}$$

Figure 2.10 shows the separation time reduction rate R_t and separation accuracy SNR of FastICA and CdICA. The latter can reduce the separation time to about 60% without losing separation accuracy, compared to FastICA. The performance of CdICA is stable for

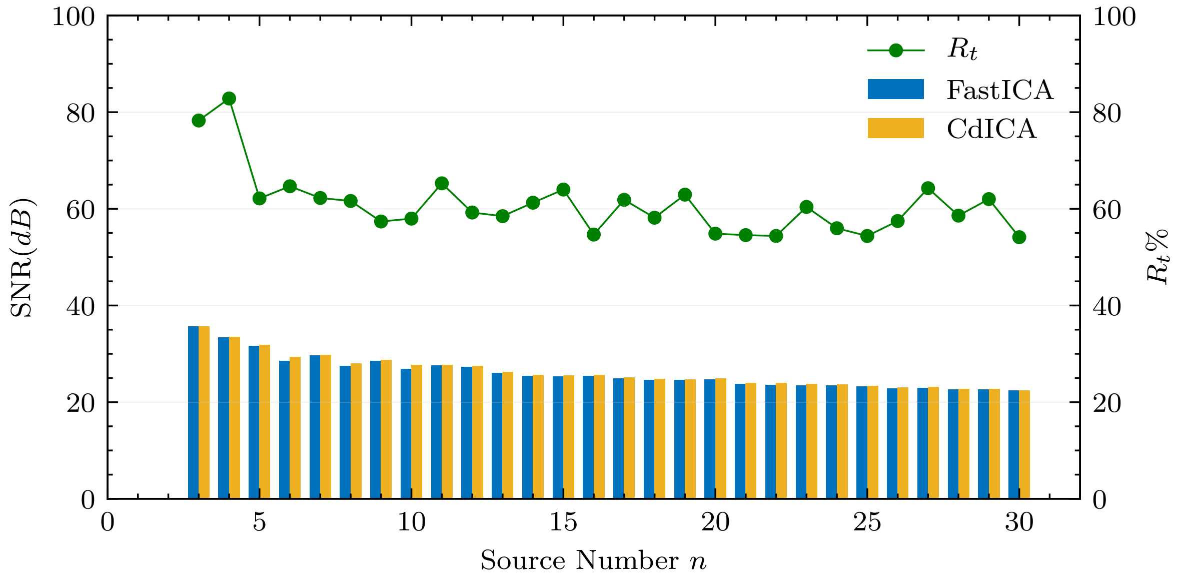


Figure 2.10: Separation time reduction rate R_t and separation accuracy SNR of FastICA and CdICA.

large n. Therefore, CdICA could be a key technology for time-sensitive factory process monitoring on the millisecond level.

2.2.9.2 Compressible Source Separation

The CSS scheme combines BSS to separate the mixed sound of several working machines and compressed sensing for reducing the amount of data transmitted over the network for partially correlated data sources. It balances between the amount of transmitted data and the separation quality. Benefiting from compressed sensing, the network throughput is optimized for the given value of desired BSS separation quality.

A total of 300 factory acoustic signals from the GADS and Factory Noise Data Set (FNDS) are used as source signals for testing. For each test, we randomly select $n \in \{5, 200\}$ original sources $\mathbf{S}$ to show the compatibility of the CSS to both small and a large number of sources. Each source component $\mathbf{s}_i$ was a 0.01-second signal with $32KHz$ sampling rate, thus the amount of data was up to $n \times m = 64000$ as a massive data scenario. The input mixtures $\mathbf{X}$ were generated with a randomly mixing matrix $\mathbf{X}$ following eq. (2.27). The mixtures $\mathbf{X}$ were compressed and reconstructed by compressed sensing through the convex optimization. The amount of transmitted data is adjusted by the data compression rate R_c, which is the ratio of compressed data by compressed sensing to the original input mixtures. Through the selection of R_c, the information

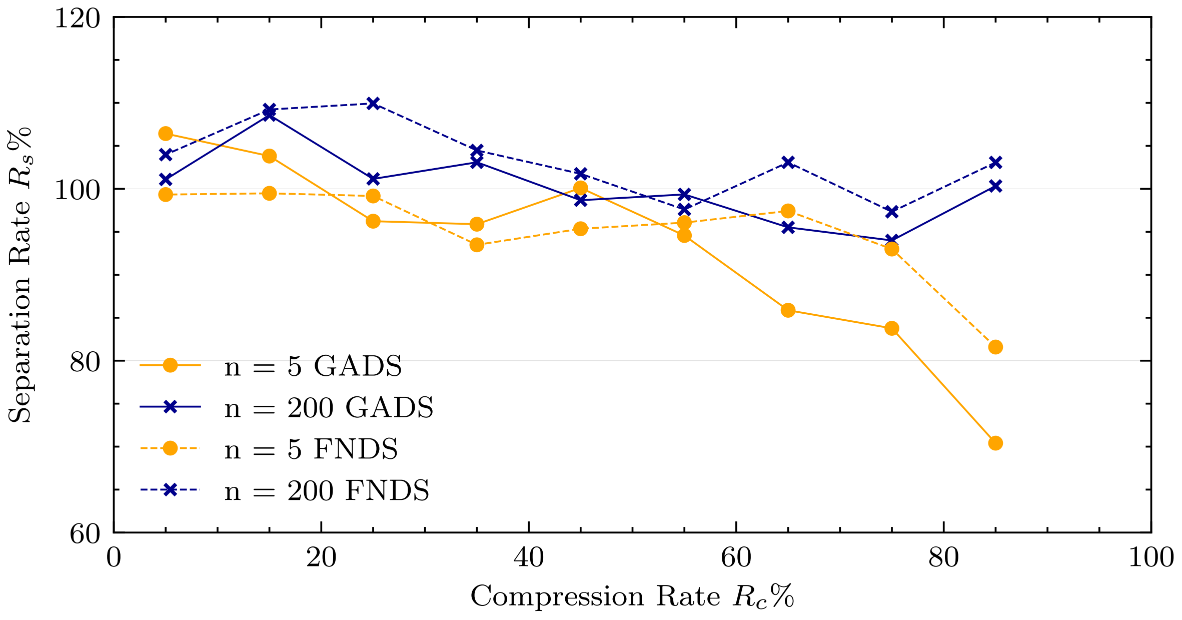


Figure 2.11: Data compression rate R_c and separation rate R_s.

losses vary due to the compression. The impairment of the separation quality comparing to FastICA can be evaluated as separation rate R_s, defined as the ratio of the CSS SNR and the FastICA SNR.

$$R_c = (1 - \frac{\|X_{comp}\|}{\|X\|}) \times 100\%. \tag{2.29}$$

Reduction rate of separation quality R_s, due to data compression:

$$R_s = \frac{SNR_{CSS}}{SNR_{FastICA}} \times 100\%. \tag{2.30}$$

Figure 2.11 shows that a large amount of data transmission is dropped, with a slight penalty on the separation quality. It is worth noting that the separation quality remains around $R_s = 100\%$ when the $R_c < 50\%$ data is compressed. Along with the increased source number n, the performance of the CSS scheme remains higher than $R_s = 90\%$ compared to FastICA, showing its suitability for large numbers of sources. In practical applications, this trade-off can be employed to achieve a certain separation quality for a given maximum amount of transferred data.

2.3 Conclusions

This chapter gave a detailed introduction to the major innovation techniques, namely network coding and compressed sensing, that we consider employing in this dissertation for reducing the padding overhead of heterogeneous data packets as well as the correlations, respectively.

RLNC is a powerful rateless coding technique that allows store-code-and-forward instead of the conventional wisdom of store-and-forward. Additionally, it enables the enhancement of throughput and robustness, as well as the reduction of energy consumption and delays. We proposed a detailed overview for RLNC, which is planned to be further extended for a survey/tutorial review submission. We believe that our classification is unique and covers all the RLNC variants in the literature.

On the other hand, we proposed a simplified compressed sensing overview that was originally published in [2] in order to highlight its importance in many communication fields and stressed the basic conditions that it needs to fulfill for accurate data reconstruction, namely *sparsity* and *incoherence* [121]. This technique allows for dramatically reducing the size/dimensionality of signals without being damaged when reconstructed. For that, it has shown to be a powerful plugin for sensors as it drastically reduces the size of the measured events, extends the battery life of constrained devices, and supports asymmetrical processing where most of the workload is shifted to the decoder. What was not explicitly reported in the literature is that compressed sensing could be applied in any field that involves large vectors. Furthermore, we proposed a blind source separation and compressed sensing approach that has the potential of leveraging anomaly detection in factory floors with low delays without the need for human intervention.

Despite the current technological advances, it remains challenging to adopt compressed sensing and network coding in some applications, such as time-sensitive and power-constrained applications. For instance, compressed sensing cannot yet achieve the 1-ms delay requirement for some Fifth Generation of mobile communication system (5G) applications, and network coding-decoding complexity remains an open issue despite the advocated variants of RLNC.

The fusion of network coding and compressed sensing is reserved for Chapter 6, where we showcase the common combination approaches in the literature and discuss in detail the conditions to enhance the overall system performance when joining them in our JoComCo scheme for cluster-based WSNs.

Chapter 3 Padding Overhead Characterization

In network coding and other FEC codes, it is commonly assumed that combined and transmitted packets have equal sizes. This is a best-case scenario, which allows the isolation of the coding benefits and policies that employ it without having a close look at the effect of individual data streams. It is therefore important to investigate whether this assumption has a valid impact on the technique's performance. Tackling this issue requires first an in-depth understanding of the real-life packet size distributions.

In this chapter, we provide an exhaustive characterization of various real-life packet traces from IP-core to video packets. We quantitatively prove that they can be irregularly distributed over the available maximum packet size and that they could considerably vary packet-by-packet. Due to the fact that the global IP traffic grew three-fold from 2015 until 2020, which makes monthly traffic to 194.4 Exabytes [20], it has become extremely crucial to pay attention to the transmitted padding overhead when performing linear coding techniques. Therefore, we evaluate and quantify this padding overhead for conventional RLNC in point-to-point and serially connected multi-hop topologies. Finally, we present and discuss the effectiveness of the available state-of-the-art solutions to zero-padding, as well as their computational complexities.

3.1 Characterization of Packet Size Distributions

We analyze the packet size distributions using large sets of anonymous passive real-life traces captured in 2015 and provided by CAIDA [152, 153]. We focus on TCP and UDP as they are the most used transport protocols. We also carry out a measurement campaign using a set of video traces from the online library of Arizona State University [13, 154].

This includes different video codecs and qualities, from the exceptional 4K video standard to the low quality Scalable Video Coding (SVC) standard.

3.1.1 IP-Core Packets

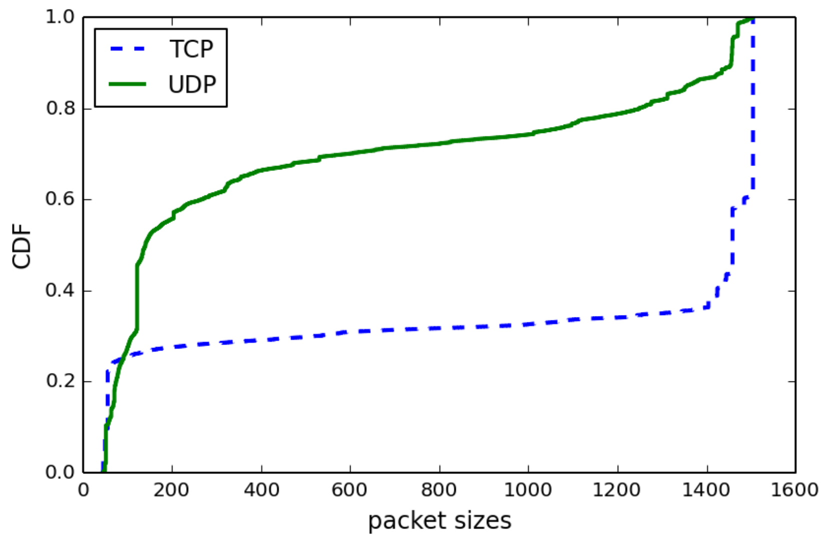


Figure 3.1: CDF of IP-based packet sizes in bytes per packet of the benchmark CAIDA trace of 70×10^5 packets [3].

CAIDA sets yearly anonymized passive traces with real Internet packets that we consider in this dissertation. As a matter of fact, it was confirmed that the packet size distribution is bi-modal with large portions of 40 and 1500 bytes, and uniformly distributed variable sizes in between [155, 156]. It is known that most Internet traffic is transported via the TCP protocol. However, now that the UDP is adopted as a transport protocol for Peer-to-Peer (P2P) protocols and streaming applications, it has become crucial to study both IP traffics.

Figure 3.1 depicts the CDF of the packet sizes for TCP and UDP protocols using a randomly selected CAIDA trace. It contains around 70×10^5 packets, which are dominated by 77.66% of TCP packets and 21.58% of UDP packets. It exposes the bimodality of the packet size distributions for TCP and UDP and shows two strong modes at 40 bytes and 1500 bytes, but with different proportions for each protocol. As for the TCP distribution, creating a generation from the trace flows is expected to contain packets of sizes 1400 bytes and 1500 bytes, with a probability close to 0.65, along with random smaller packets. According to the UDP distribution, this is different as the generations are expected to have more small packets, with a probability of around 0.5.

3.1.2 Video Packets

As a result of the substantial growth in mobile video traffic, nearly a million minutes of video content was expected to cross the network every second by 2019, based on the predictions of the Cisco VNI [157]. Moreover, visual websites such as Dailymotion and YouTube, as well as video calling applications, including Whatsapp and Zoom, encounter an exponential increase in the number of video downloads and uploads. Not only their video exchanges escalated, but also the consumers' demand for higher quality videos as the performances of their systems have increased. According to the Cisco statistics, the IP video traffic reached 82% of all IP traffic in 2020, while it used to be around 70% in 2015 [20]. As for business IP video traffic, it has reached 64% of the business IP traffic of 2020, after being just 42% is 2015 [20]. Generally, there are two broad types of video models, namely conversational videos including conferencing applications and full-motion videos, such as movies, TV shows, and sports events. Nevertheless, we will more or less focus on the full-motion video types, as they display a wide range of scenes with varying levels of activity. This includes background and foreground with scene changes as in video surveillance, sports broadcast, and movies. The global fluctuation of activities generates a set of Variable Bit Rate (VBR) sequences when the video is encoded using modern efficient video coding standards [158–163]. The varying degree of movement produces frames with different sizes (in bytes), which in turn results in varying sizes of the packets that carry the encoded data over multimedia networks. As a result, studying video data size distributions and their impact on the network coding padding overhead is interesting. For this purpose, we selected a collection of five representative full video traces of long full-motion videos that were encoded using various video coding techniques from the publically available Arizona State University online library [154]. The library has different encoding/decoding standards [164] resolutions, ranging from the low resolution CIF to the 4k format, as detailed in Table 3.1.

Despite the fact that universal Ethernet Maximum Transmission Unit (MTU) is around 1500 bytes, video frames can grow up to more than 10 Kbytes due to their nature [165, 166]. With the aim of ensuring their transmission, they should undergo fragmentation and a packetization step to meet the MTU constraint. Consequently, an entire frame of high definition video might require 100 IP packets in order to carry all of its elementary stream packets.

Figure 3.2 depicts the CDFs of the video packet sizes that correspond to the video traces in Table 3.1. We observe that video traces have a quite diverse packet size distributions. However, they have a common strong mode at 1500 bytes, which is

Table 3.1: Characteristics of the benchmark video traces from the online library of Arizona State University [12, 13] with 30 FPS for CIF resolution video and 24 fps for the remaining videos [4].

Video Label: Enc. Appr. (Resol. Format)	Resolution [pixels × pixels]	Number of Video Frames	Number of Resulting Packets
SVC (CIF)	352 × 288	289040	336345
VP9 (HD)	1920 × 816	17592	45388
SVC (HD)	1920 × 1080	86268	336345
H.264 (HD)	1920 × 816	17592	91551
H.265 (4k)	4096 × 1744	17592	163921

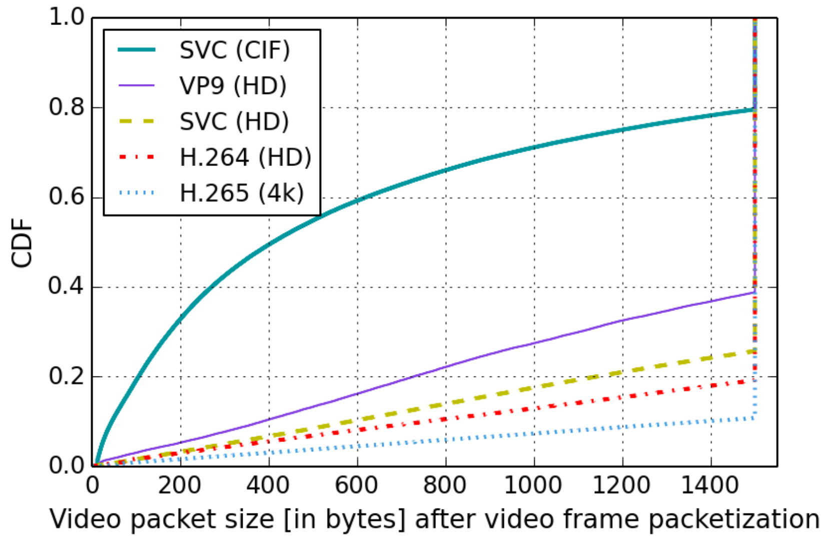


Figure 3.2: CDF of video packet sizes in Bytes per packet of the benchmark traces [4].

expected since it is the predefined MTU when chunking a video frame. Additionally, videos with higher resolutions, i.e which have larger frames, such as the 4k video trace, have a larger probability of being fragmented into packets of the MTU size compared with others. For instance, only around 20% of the SVC (CIF) video packets have 1500 bytes, whilst approximately 90% of the H.265 (4k) video packets have 1500 bytes.

3.2 Padding Overhead in Generation-Based RLNC

In order to reduce the complexity that the technique of network coding brought due to the need for centralized knowledge of the graph topology, Chou et al. introduced a practical solution to handle it [57], which consists of performing network coding on sets of packets called generations (also called chunks, batches, classes, etc.). The size of these generations highly depends on the type of application, e.g., video streaming and file transmission. Accordingly, employing generation-based network coding should not bring delays, overhead and extra computational complexity [167, 168].

On the other hand, linear codes for FEC, including RLNC, come at the cost of creating different kinds of overhead. Generally, there exist three common types that we could counter when using RLNC, namely *coding vector*, *linear dependent packets*, and *padding* overheads. The first refers to the size of the encoding vector, which is constituted by the number of coding coefficients. The second depends on the finite field size when linearly dependent packets are generated and it represents the extra transmissions. The third is created due to the fact that the linear combinations can only be created using entities of the same size. The former two types of overhead were covered in lots of studies, and since they are highly controlled by the field size, an appropriate finite field size choice is sufficient to minimize them. Besides, they can be predicted and estimated in advance.

In this section, we exclusively focus on the padding overhead, as it is treated the least from the research community compared to the other forms of overhead.

3.2.1 Over a Point-to-Point Topology

We consider the case of a basic point-to-point transmission of generations of data packets using RLNC. Each generation to transmit is composed of N consecutive packets of a given source packets sequence $P_1, P_2, ..., P_N$. We denote their sizes by $L_1, L_2, ..., L_N$ respectively, in symbols of the GF(q), i.e., $\log_2(q)$ bits per symbol. In this chapter, we stick with bytes for simplicity. The largest packets in the generation have the size $L_{\max}$, defined as $L_{\max} = \max\limits_{1 \leqslant i \leqslant N} L_i$.

3.2.1.1 Padding Overhead Characterization

In preparation for the encoding process, the sender needs to fill the packets that are smaller than the largest ones of size $L_{\max}$ inside the generation with trailing zero symbols. We refer to these trailing symbols as the *zero-padding overhead* (see Figure 3.3). The sender then encodes the generation using random linear combinations of these full-size packets obtained by drawing random finite field coefficients.

When using block codes, such as generation-based RLNC, the sender usually forms the generation after acquiring a priori knowledge of the packets, which is used for generating coded packets. In most cases, the packets to combine are already available at the sender's

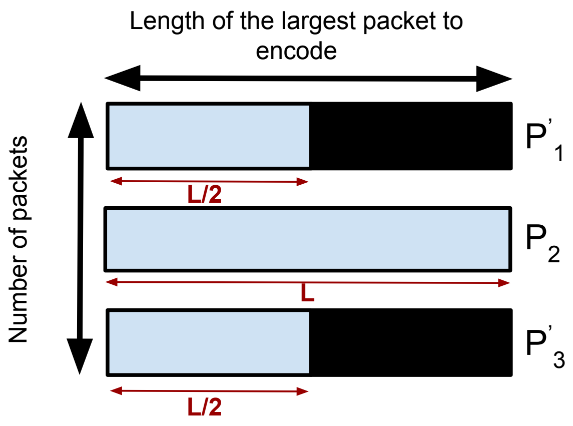


Figure 3.3: An example of a three packet generation, with original sizes $L/2$, L and $L/2$ respectively. If coding with the largest packet in the generation is performed, it will result in transmitting at least an overhead (represented by the black space) that corresponds to 50% of the initial data to transmit. We are implicitly sending one extra packet in this case.

buffer, i.e. their sizes are already known. Therefore, the problem of unknown traffic size can be eliminated principally if the node adapts dynamically to this size change. Regardless of the original sizes, the padding overhead always depends on the size of the largest packet, since each packet is padded out to the size $L_{\max}$, as well as the sum of all the packet sizes present in the generation. We calculate the zero-padding overhead created and carried within every generation regardless of the coding approach employed as follows [3]:

$$O_G = \sum_{i=1}^{N}(L_{max} - L_i) \times (\sum_{i=1}^{N} L_i)^{-1} \tag{3.1}$$

where L_{max} represents the size of the largest packet in the generation, and L_i is the size of P_i inside one generation. Additionally, the packet size distribution cannot say much about the overhead in general, because the generation depends only on the consecutive packets in a flow.

3.2.1.2 Numerical Results

We previously showed that packet sizes present a high level of randomness. Additionally, we proved analytically that performing generation-based RLNC on real-life packets comes at the cost of creating padding overhead. To further look into it, we investigate its minimal magnitude as well its occurrence using the same aforementioned traces in terms of the generation size.

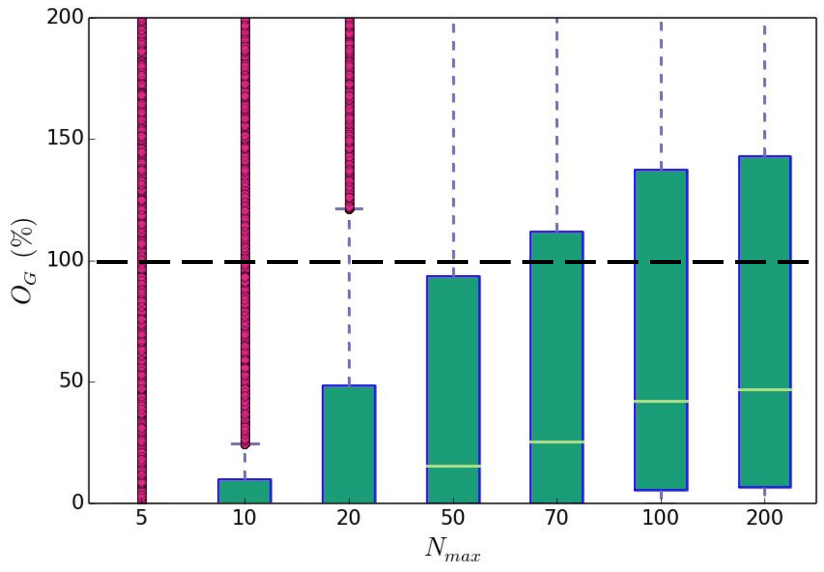
3.2.1.2.1 IP-core Packets First of all, we define N_{max} as the number of packets in a unidirectional flow that should not exceed the default generation size N. As a matter of fact, packets in some flows in the CAIDA traces are used up before filling the generation of size N. Nevertheless, we decided to keep them in the simulation procedure to remain faithful to the real-life traces. For example, $N_{max} = 50$ encompasses all the possible maximum generation sizes that are equal or less than 50, depending on the unidirectional flow. During the encoding process, we filled the smaller packets with trailing zeros before performing the standard RLNC operations as detailed in Chapter 2.

In the following, we use box plots for an accurate statistical overview of the padding overhead. We remind that the boxes give the median, as well as the upper and lower quartiles. The whiskers give the 10% and the 90% quartiles.

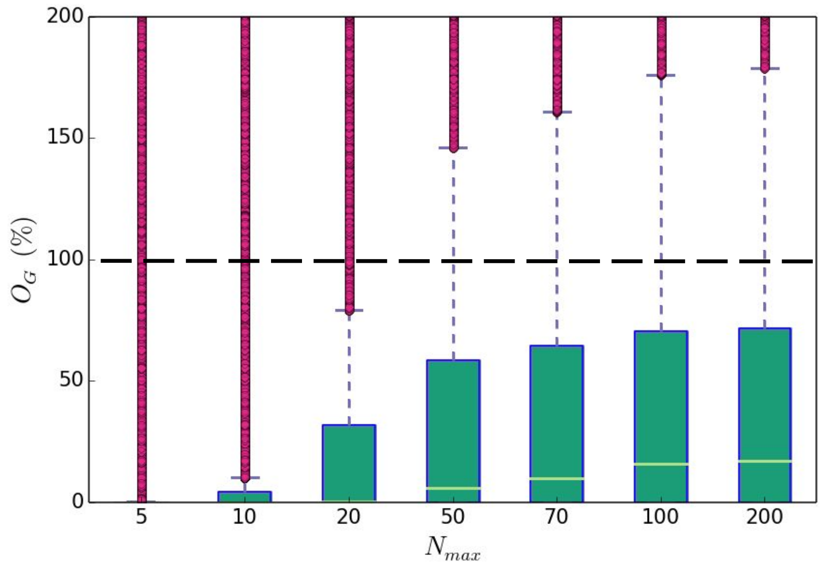
Figure 3.4 depicts the padding overhead O_G as a function of the generation size N_{max} for the TCP and UDP traces that we consider in this chapter. For small generation sizes, such as $N_{max} = 5$, we observe an overhead dominated by an enormously random outlier that surpasses 200% of the original packets inside such small generations. For higher TCP generation sizes in Figure 3.4a, we observe that the padding overhead increases sharply and remarkably exceeding the amount of input data when the generation size increases. For example, it has a third quartile close to 140% for $N_{max} = 200$. As for the UDP trace, Figure 3.4b shows that for larger generation sizes, the padding overhead rises as well. Even so, it has less significant overhead compared to the TCP generations. For instance, the third quartile for large generations never exceeds 70%, and the median is located around 20%.

Overall, because of the mingled minimum and the first quartile at 0%, we cannot estimate or predict the number of generations that are not susceptible to carrying padding overhead. As a result, we complement our numerical results with Figure 3.5, in order to highlight the percentage of generations that exactly contain unequal packet sizes, i.e., inevitably create padding overhead when performing RLNC.

Figure 3.5 shows the presence of padding overhead when performing RLNC on TCP and UDP packets in percentages as a function of the generation size. We observe that a CAIDA trace contains at least 20% and 29% of the generations have padding overhead for the TCP and UDP respectively. This overhead rises exponentially when the generation size increases. For example, around 85% of the TCP data transmitted in a generation of size $N_{max} = 200$ contains trailing zeros. As for its presence in UDP-based generations



(a) TCP



(b) UDP

Figure 3.4: Percentage of minimum zero-padding overhead carried for different generation sizes of a CAIDA packet trace [3].

when using RLNC, the padding overhead presence converges around 68% for generations with $N_{max} = 150$ and above. It remains, however, alarming to observe the presence of padding overhead, regardless of the generation size. Such numbers are expected because of the packet size distributions displayed previously in Figure 3.1, where around 70% of the packets have sizes between 1400 and 1500 bytes, and most of the remaining are around

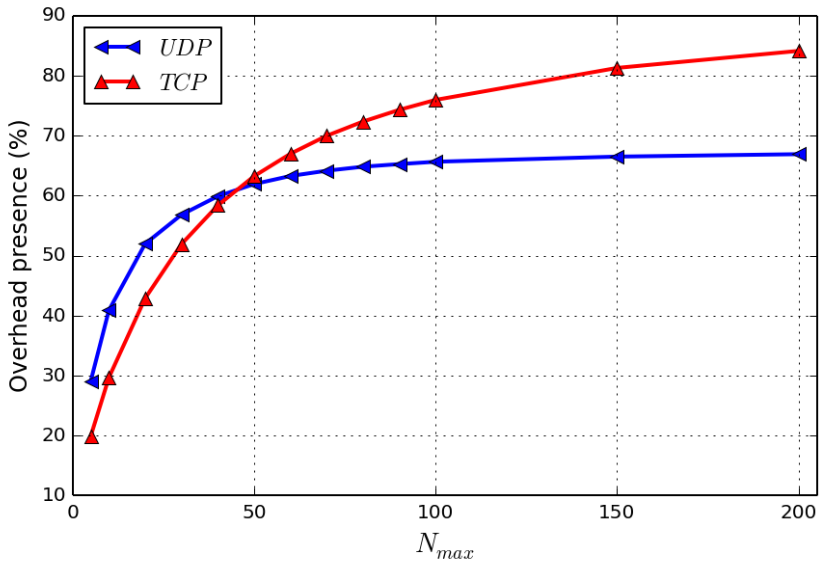
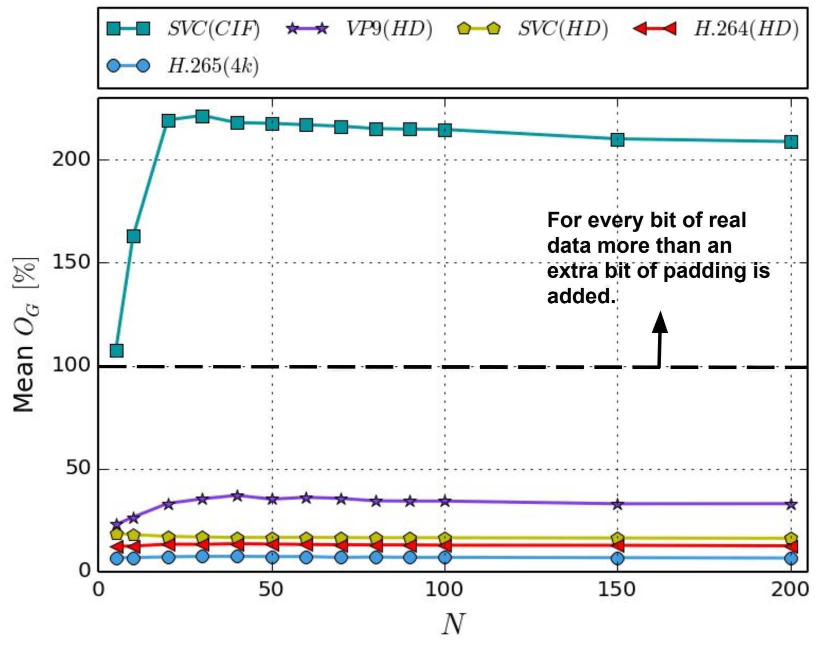


Figure 3.5: Percentage of the presence of zero-padding overhead as a function of the generations of the benchmark CAIDA packets trace [3].

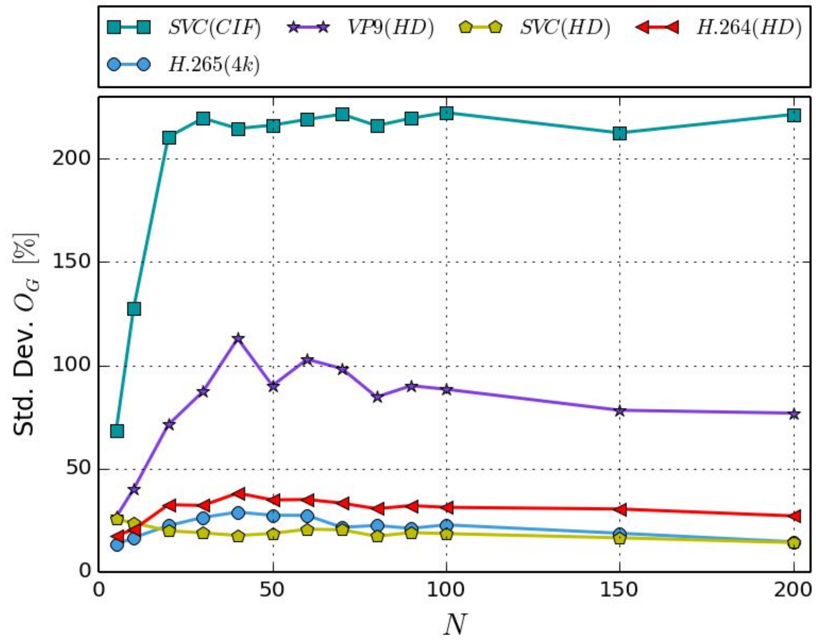
40 bytes. Various statistical factors demonstrate the potential of large TCP generations in carrying enormous padding overhead. From the Internet traffic classification point of view, it is due to the fact that the TCP protocol generally supports a wider range of Internet applications.

3.2.1.2.2 Video Packets Figure 3.6 illustrates the means and standard deviations percentages of the zero-padding overhead O_G carried inside a generation compared to the size of the original packets as a function of the generation size. We recognize that the generations of the CIF video format include on average a huge padding overhead, which is of a size that is more than double the original data itself. For generations of sizes $N = 20$ and above, the padding overhead already exceeds 200%, which implies that the video data is tripled. This extravagant padding overhead is due to the highly variable frame sizes with many small frames, as displayed in Figure 3.2. Furthermore, we observe that higher resolution videos tend to have less padding overhead. For example, for $N = 50$, the padding overhead of the VP9 (HD) video has a mean around 35%, as well as a median of 20%, whereas, for the H.265 (4k) video, it has a mean of 7% and a median around 3.8%.

For in-depth insights, Figure 3.7 and Figure 3.8 depict box plots of the padding overhead for the VP9 and the H.265 video traces. We remark in Figure 3.7 that the generations' padding overhead have medians around 20%, and means around 30-35%.



(a) Mean of Padding Overhead O_G



(b) Standard Deviation of Padding Overhead O_G

Figure 3.6: Means and standard deviations of padding overhead O_G [in percentage] as a function of the generation size N [3].

This is due to its high proportion of small packets, as displayed in Figure 3.2. However, for Figure 3.8, the median overheads are around 3-4%, and means are around 6-7%, which

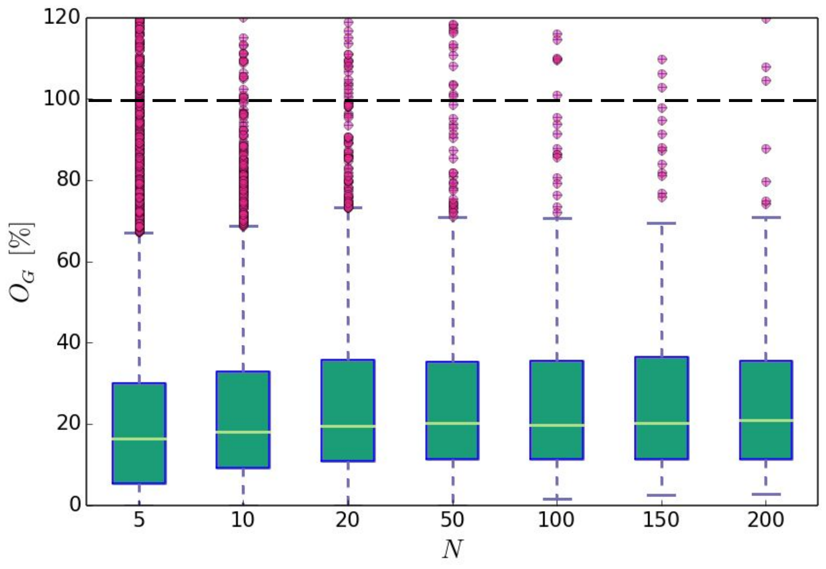


Figure 3.7: Box plots of zero-padding overhead O_G as a function of the generation size N for the VP9 (HD) video trace.

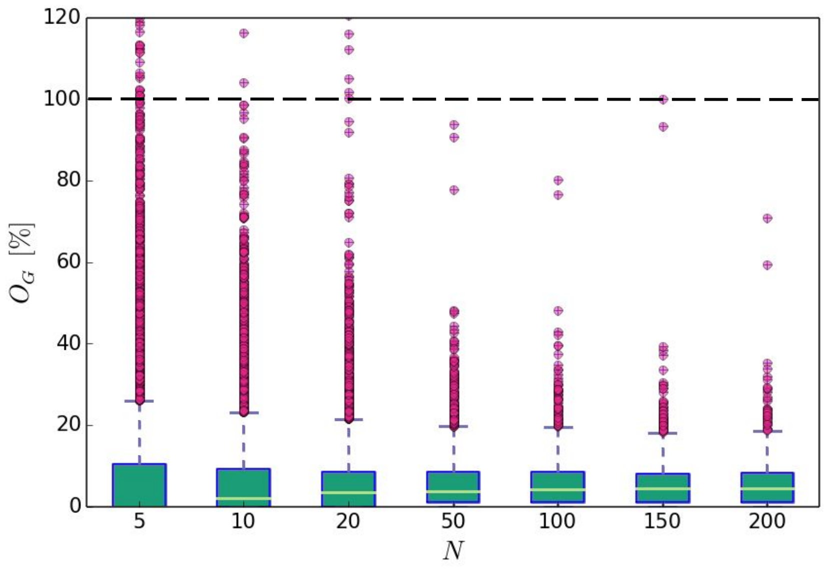


Figure 3.8: Box plots of zero-padding overhead O_G as a function of the generation size N for the H.265 (4k) video trace [3].

is not as high as the VP9 (HD) video, due to its large size frames by default. Additionally, both Figures show a high occurrence of outliers for small generation sizes such as $N = 5$, compared to large size generations, e.g., $N = 200$. This means that these generations are susceptible to comprising very large padding overheads. As a matter of fact, small generations only “sample” short portions of the video frames sequence. When we consider only a few consecutive encoded video frames, there is actually a potential for immensely different frame sizes. For example, when the frames include a large intra-coded I-frame and some small bidirectionally B-frame predicted frames of the Group of Pictures (GoP) structure of the coding [12].

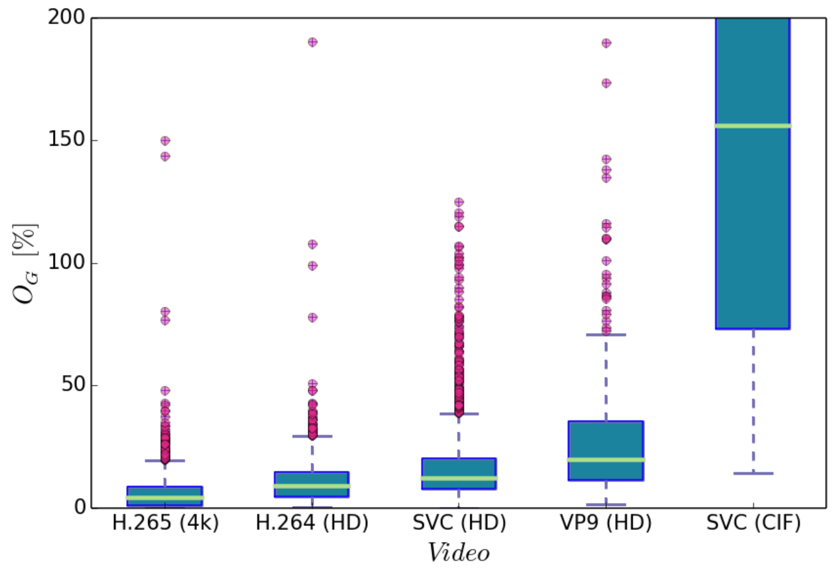


Figure 3.9: Percentage of minimum padding overhead for a collection of video traces for $N = 100$ [4].

Finally, Figure 3.9 shows the box plots of the zero-padding overhead from all the videos in Table 3.1, for the generation size $N = 100$. We notice that the higher the video resolution, the lower the padding overhead it tends to have. This is due to the fact that video frames for high-resolution videos are quite large, thus requiring a large proportion of the MTU-size packets. However, their outliers are mostly present between 20% and 50%. Consequently, generation-based network coding comes at the expense of a substantial padding overhead, despite the fact that video frames are usually large and require a substantial number of full-size packets for their transmission.

3.2.2 Over an ℓ-Hops Topology

Although the padding overhead is usually canceled out by the decoder due to the fact that the delivered data should not grow and maintain its initial size, the coding systems still put additional computational effort into processing both at the source and the destination, without neglecting the fact that it is uselessly carried across the channel. This investment and practice adds up to the overall energy consumption, and it constitutes a real handicap for the communication systems that intend to become more efficient and cope with the 5G standard. In scenarios where larger network topologies are used, this padding overhead could become a bigger burden if it is transmitted over and over with every encountered recoder. On the other hand, it is known that RLNC has the salient feature of enabling intermediate nodes to recode the received coded packets, unlike other FEC codes.

We focus on a portion of a wireless mesh network, that only consists of ℓ-hops, to persuade the reader about the effect of losses as well as the incurring overhead on RLNC performance. Moreover, we only consider high field sizes in order to emphasize the principal contributions independently with the linear dependency overhead without loss of generality to smaller finite fields.

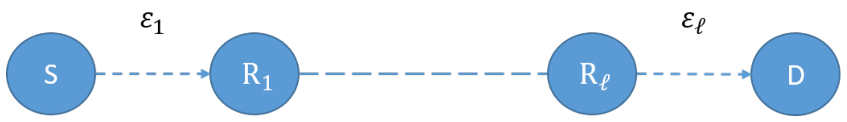


Figure 3.10: An example of an ℓ-hop line network with loss rate ε_i per link, $i \in \{1, \cdots, \ell\}$.

3.2.2.1 Characterization

We consider, similarly to the previous setups, the transmission of a generation composed of N packets through an ℓ-hop wireless network similar to the one depicted in Figure 3.10. We recall that this scenario is also valid for topologies with similar settings, such as wireless mesh networks. We suppose that we propose to transmit N packets, $P_1, \cdots, P_N$ of sizes $L_1, \cdots, L_N$ respectively and in units of symbols of the finite field specification. The size of the largest packet in the generation is denoted as $L_{max} = \max\limits_{1 \leqslant i \leqslant N} L_i$.

We start by defining the minimum effective payload that is sent in an ℓ-hop network by $\mathcal{E}_N^\ell$, as follows:

$$\mathcal{E}_N^\ell = \sum_{j=1}^{\ell} \sum_{i=1}^{N} \frac{L_i}{1 - \varepsilon_j} \quad \text{(bytes)}, \tag{3.2}$$

where, ε_j represents the loss rate for the link j, $j = 1, 2, \cdots$.

Since RLNC is a compute-and-forward scheme, it fulfills the rule of the min-cut max-flow, i.e., the probability of successful delivery between the sender and the receiver nodes is $\min\limits_{1 \leqslant j \leqslant \ell} \{(1 - \varepsilon_j)\}$. However, for standard routing and other coding approaches, such as Luby Transform (LT), Reed Solomon, Raptor, and Low Density Parity Check (LDPC) codes, they are only capable of store-and-forward. This makes them vulnerable to losses in such network topologies. Their probability of successful delivery can be expressed as $\prod_{j=1}^{\ell}(1-\varepsilon_j)$, which diverges from the aforementioned end-to-end delivery probability of network coding. According to these conditions, in total we send with standard RLNC in an ℓ-hop network:

$$\mathcal{T}_N^\ell = \sum_{j=1}^{\ell} \left\lceil \frac{NL_{max}}{1-\varepsilon_j} \right\rceil \quad \text{(bytes)}. \tag{3.3}$$

Eventually, we define the total padding overhead, in bytes, created by RLNC encoders in ℓ-hop network, O_N^ℓ, as follows:

$$\begin{aligned} O_N^\ell &= \mathcal{T}_N^\ell - \mathcal{E}_N^\ell & (3.4)\\ &= \sum_{j=1}^{\ell}\left(\left\lceil \frac{NL_{max}}{1-\varepsilon_j}\right\rceil - \sum_{i=1}^{N}\frac{L_i}{1-\varepsilon_j}\right). & (3.5)\end{aligned}$$

3.2.2.2 Numerical Results

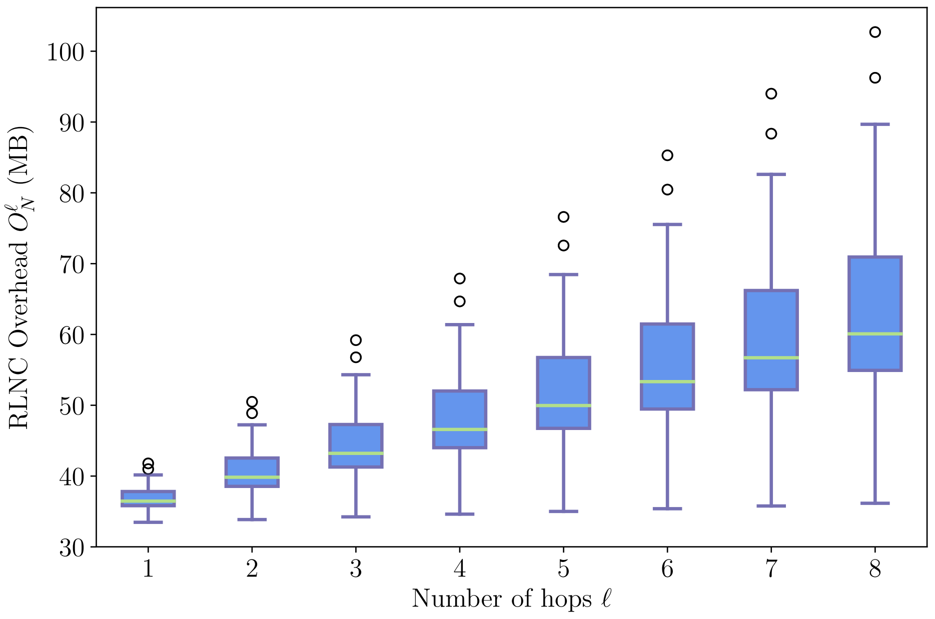


Figure 3.11: Box plots of padding overhead O_N^ℓ in megabits for a TCP trace, as a function of the number of hops ℓ, with $N = 32$, $\ell = 8$, $\varepsilon_1 = 0.4$, and all other links are supposed to be perfect.

Figure 3.11 illustrates the overall padding overhead in an ℓ-hop network, where the first link has a loss probability $\varepsilon_1 = 0.4$. The remainder of the links is assumed to be loss-free. We notice that such padding overhead increases exponentially with the number of hops. We used a TCP trace from CAIDA containing $12 \cdot 10^3$ packets. We observe that we have transmitted in total 50% of the coded data as trailing zeros once by the fifth hop in the network. This continues to climb as the number of hops increases, despite the fact that no losses or erasures occur. After a certain number of hops, the padding overhead will simply surpass the meaningful coded data being conveyed. We conclude that RLNC unfortunately loses its statelessness and behaves similarly to the traditional linear codes, which have no control over the losses or the overhead as the number of hops increases, except that here RLNC is unable to deal with the propagation of the padding overhead.

3.3 Existing Padding Reduction Approaches

To the best of our knowledge, the padding overhead issue in RLNC for the distribution of common IP-core packets and VBR encoded media has only received limited attention to date, despite its aggravation in large multi-hop networks especially, as previously discussed. Compta et al. proposed in 2015 four padding overhead reduction approaches that aim to pack the unequal-sized packets into fixed size packets, which are amenable to conventional RLNC [169]:

1. *Simple fragmentation*
2. *Fragmentation and bundling*
3. *Simple bundling*
4. *Chain and fragmentation*

They can be grouped into two categories based on *bundling* and *fragmentation*, before finishing with padding to the MTU size. Two of the suggested solutions showed a significant reduction, which is below 5% when tested on video packets from Arizona State University online traces [169]. Since these schemes are RLNC related approaches, we propose to encode and decode the resulting generations similarly to the ones with RLNC, in order to have a fair comparison. For instance, we propose the usage of the Gaussian elimination algorithm for decoding in all the approaches. In the following, we briefly describe the schemes and discuss their effectiveness through their computational complexities.

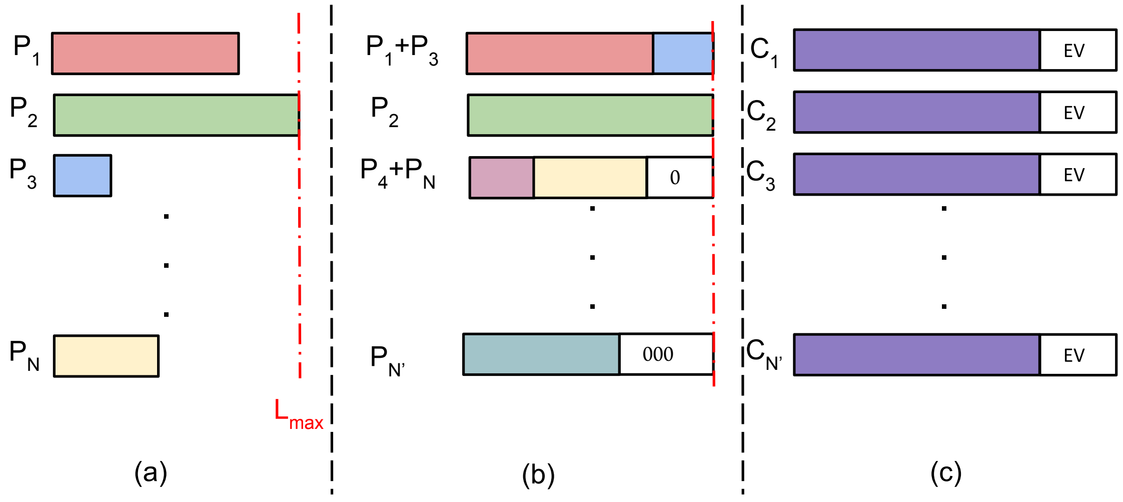


Figure 3.12: Simple bundling scheme example for a generation of N original unequal-sized packets.

3.3.1 Simple Bundling Scheme

Such a scheme relies on finding the best possible bundling solution that allows the filling of the gaps in smaller packets using other small ones to reduce or, if applicable, eliminate the need for padding. The encoding is later performed on the resulting N_{SB} new packets, $N_{\mathrm{SB}} \leqslant N$. This procedure is similar to a bin packing problem, which strives to pack different sized objects (here packets) into the minimal number of containers (packets) of the same volume (fixed packet size). This turns out to be an NP-hard problem with high computational complexity. Nonetheless, other bundling strategies seem to alleviate it while compromising on padding. Figure 3.12 illustrates a basic example of how the small packets are being packed together without exceeding the size of the largest packet in the generation (here P_2). We notice that the zero-padding is indeed inevitable in this example to match with the generation's largest packets. Therefore, it cannot always guarantee a solution to the issue. For example, with the *simple bundling* scheme, the padding overhead drops from 100% to 10% according to the benchmark video trace portion used in [169].

Inspired by the *bin packing* idea, we would like to find the best and valid arrangement to store the packets of a generation with the least padding overhead $\mathcal{O}_j$ that incurs in every newly created packet P'_j. These packets have size $L_{\max}$ each, which corresponds to the largest original packet size in the original generation. Moreover, we denote the total zero-padding overhead by $\mathcal{O}_{SB}$, and the resulting new generation size by N_{SB}.

Consequently, the problem can be formulated as:

$$\min \sum_{j=1}^{N_{SB}} \zeta_j \equiv \min \sum_{j=1}^{N_{SB}} (L_{\max} - \sum_{i=1}^{N_{SB}} L_i \delta_{ij}) \equiv \min \sum_{j=1}^{N_{SB}} \mathcal{O}_j \tag{3.6}$$

subject to:

$$\sum_{i=1}^{N_{SB}} L_i \delta_{ij} \leqslant L_{\max} \zeta_j \tag{3.7}$$

$$\sum_{i=1}^{N_{SB}} \delta_{ij} = 1 \tag{3.8}$$

where,

$$\delta_{ij} = \begin{cases} 1 & \text{if } P_i \text{ can be stored in } P'_j \\ 0 & \text{otherwise/} \end{cases} \tag{3.9}$$

The indicator function δ_{ij} informs whether P_i is stored in the j new packet, and its existence is also expressed using the indicator function ζ_j as follows:

$$\zeta_j = \begin{cases} 1 & \text{if } P'_j \text{ exists} \\ 0 & \text{otherwise.} \end{cases} \tag{3.10}$$

As a result, there are up to N_{SB} new packets P'_j, $1 \leqslant j \leqslant N_{SB} \leqslant N$ to encode. The *first fit decreasing* heuristic algorithm is an optimal working solution to the simple bundling problem. It simply consists of sorting the packets based on their sizes in decreasing order and then placing the current largest packet in the first position where it can fit [170]. This method on its own has a complexity of $\mathcal{O}(N^2)$. As a result, if the RLNC approach is applied to the newly formed generation of size N_{SB}, the total encoding complexity per generation is expected to be $\mathcal{O}(N^2 + N_{SB}^2 L_{\max})$ in the finite field. As for the decoding complexity, it is in the order of $\mathcal{O}(N_{SB}^3)$. This is similar to the decoding complexity of RLNC, since it is mainly led by the cubic factor related to the Gaussian elimination operation for a generation decoding.

3.3.2 Fragmentation Scheme

Whilst the aforementioned scheme does not allow the fragmentation of packets, the class of *fragmentation* approaches attempts to fill the previously unfillable gaps that resulted in generating the padding by defining a fragmentation size F_s for the packets before bundling them. This scheme's performance is highly dependent on F_s as well as the overall size of all packets in one generation. Figure 3.13 illustrates an example of a randomly chosen fragmentation size. We observe that the fragmentation size choice resulted in the increase in the number of packets in the generation $(N' \geqslant N)$ as well as a clear presence of the padding overhead. Instead of simply reducing the gaps generated from the bin packing by chunking the packets down to a prescribed size, it generated more packets to encode, i.e., larger encoding vector overhead, a higher encoding/decoding complexities as well as a padding overhead due to the filling of the gaps that could not be filled with the fragments.

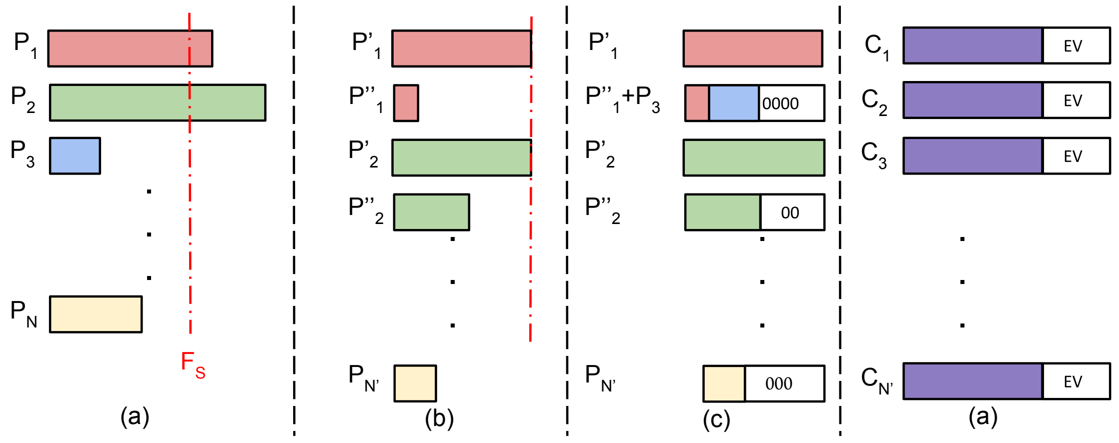


Figure 3.13: Simple fragmentation example.

Additionally, this procedure incurs a signaling overhead due to the entire reconstruction of all packets. As much as it seems simple and capable of extremely reducing the zero-padding overhead, it literally exchanges it with this new form of signaling overhead, which we denote by $\mathcal{O}_F$.

As a result, F_S has to be carefully tuned to avoid the aforementioned issue. The simultaneous reduction of the padding and the fragmentation overhead is a paramount for such a scheme. We express it using the multi-objective optimization problem as follows:

$$\min \mathcal{O}_F \tag{3.11}$$

$$\min \#F_s \tag{3.12}$$

subject to:

$$\sum_{i=1}^{N_F} L'_i \delta_{ij} \leqslant F_s \zeta_j \quad (3.13)$$

$$\sum_{i=1}^{N_F} \delta_{ij} = 1 \quad (3.14)$$

$$F_s \leqslant \mathcal{C}_{\text{MTU}}, \quad (3.15)$$

where L'_i is the size of the data filled inside the new packet P'_i, and N_F is the new generation size after fragmentation. Being more interested in decreasing the padding overhead, we could decide it to be the dominating function of the optimization problem. This is referred to as the *chain and fragmentation* scheme, which consists of concatenating the packets together back-to-back into one long string then deciding the appropriate fragmentation size, i.e., the new packet size, denoted F_{CF} [169], before fragmenting the long string into fixed size packets. Hence, the problem could be rewritten as:

$$\text{Find} \;\; F_{CF} \in \mathbb{N}^* \;\; : \;\; F_{CF} \wedge (\sum_{i=1}^{N} L_i + \mathcal{O}_{CF}), \;\; F_{CF} \leqslant \mathcal{C}_{\text{MTU}}. \quad (3.16)$$

Thus, $\mathcal{O}_{CF} \neq 0$ iff $F_{CF} \nmid \sum_{i=1}^{N} L_i$. Furthermore, this is a linear problem that does not instigate a higher complexity to the encoding process, which is in the order of $\mathcal{O}(F_{CF} N_{CF}^2)$. As for decoding, it requires $\mathcal{O}(N_{CF}^3)$ operations in the finite field to recover the generation. With the *chaining and fragmenting* scheme, the excess data to be transmitted decreases logarithmically when F_S increases. Ultimately, for high F_S values (with $\max\{F_s\} = \mathcal{C}_{MTU}$), the zero padding becomes minimal compared to the other approaches (including RLNC).

3.4 Conclusions

In this chapter, we examined the overhead that arises from the zero-padding of variable size packets, including packets of VBR videos and IP packets, for reliable transmission with RLNC, and showed its strong connection with the real-life packet size distributions. Our numerical results revealed that padding the small packets with trailing zeros to the maximum packet size generates an overhead on the order of $20\% - 50\%$ or more for the benchmark video traces. Furthermore, TCP traffic could, on average, have an overhead equivalent to 75% of the total data transmitted, i.e., three out of four bits are overhead. State-of-the-art padding overhead reduction approaches proposed to pack the unequal size packet into fixed-sized packets, through bundling or simply chaining and fragmenting. The chain and fragmentation scheme that relies on chaining the input packets, forming bulk data to be fragmented according to a specific parameter size, dramatically reduced the padding overhead. Nevertheless, these schemes incurred a high computational complexity due to bundling and signaling overhead when fragmenting. Additionally, the performance evaluations in [169] have been limited to 320 frames of a mostly motion-free video, which is not enough to confirm the effectiveness of the proposed schemes. In the following two chapters, we proposed novel schemes that effectively deal with the padding overhead problem. Interestingly, the heterogeneity in the packet sizes enables some schemes to exploit it in a way that allows them to produce unequal-sized coded packets, and some to solemnly XOR data packets without the need for generating any coding coefficients.

Chapter 4 | Macro-symbol Based Approaches

In this chapter, we introduce an alternative to the state-of-the-art padding overhead reduction techniques, which pack the unequal-sized packets into fixed-sized ones for conventional packet granularity RLNC. For instance, we propose a different class of RLNC that operates on columns of macro-symbols (MS). These are small subsets of the packets, such as concatenated bytes, which allow the partition of the variable-size packets into fixed-size macro-symbols, while preserving the overall concept and properties of RLNC. We quantify the impact of the macro-symbol sizes on the padding overhead, as well as the computational complexity for encoding and decoding.

Relying on macro-symbols, we propose the *progressive shortening* scheme, which is the first RLNC based approach to ever generate unequal-sized coded packets, even when recoding is performed at intermediate nodes. We provide numerical results about the number of macro-symbols needed for decoding compared to the traditional RLNC, as well the throughput for encoding and decoding when implemented in Kodo, the fully-fledged network coding software library [171]. Our results show that this scheme reduces significantly the padding overhead even for small field sizes despite the penalty on the decoding throughput. Furthermore, we analytically and numerically prove that this scheme has the capability of healing the coded generations when losses and unnecessary redundant data are being transmitted to the recoder. Consequently, the padding overhead cannot be propagated, which is unlikely for unlike conventional RLNC and other coding techniques, as we showed in Chapter 3.

In order to verify that the progressive shortening can be applied with general source coding techniques, just like RLNC, we propose to show the gain obtained with Robust Header Compression version 2 (RoHCv2) in a real-life application. Consequently, our implementation results reveal a 20% payload delivery efficiency enhancement compared to standard RLNC.

Finally, we highlight that the progressive shortening approach does not only inherit recoding from conventional RLNC, but it also shows improvements when sparse coding coefficients and overlapping generations are considered. Note that the remainder of this chapter was originally published in [4–6, 31].

4.1 Macro-Symbol Concept

Macro-symbols are subsets of concatenated symbols of the packet in the finite field (e.g., bytes or bits). We propose and describe their concept and show how macro-symbol-based RLNC (MS RLNC) can mitigate the padding overhead thanks to the finer granularity that facilitates a clever way of coding and decoding.

4.1.1 Design: From Packets to Macro-Symbols

The zero-padding overhead with generation-based RLNC appears because of the packet level granularity of the common coding. Specifically, coded packets are generated in traditional RLNC by linearly combining the stack of packets that have the same size $L_{\max}$ in a generation. Our idea for reducing this type of overhead consists of performing network coding based on small subsets of the data, instead of packets inside one generation, because it gives the opportunity to seize and exploit the overhead incurred due to the heterogeneity of packet sizes. To do so, we decided to work on what we named *macro-symbols*, which are equal size sets of symbols that are present in each packet of a generation.

When deciding the macro-symbols size, it is crucial to avoid the creation of zero paddings. This is not straightforward because it only depends on the packet size distribution, which is quite random. We expect that the smaller the macro-symbols are, the least trailing zeros have to be added. Nevertheless, their sizes can therefore vary from one symbol (e.g., one byte) to the size of MTU $\mathcal{C}_{\mathrm{MTU}}$. We denote the size of macro-symbols set for all macro-symbols in a generation by μ. It is inevitable to have some padding overhead within the last macro-symbol of a packet. We also define the resulting number of macro-symbols in packet P_i as:

$$\Lambda_i = \left\lceil \frac{L_i}{\mu} \right\rceil, \tag{4.1}$$

with $\lceil a \rceil$ being the smallest integer larger or equal to a. As such P_i could be seen as a concatenation of macro-symbols $s_{i1}, \cdots, s_{i\lambda}, \cdots, s_{i\Lambda_i}$.

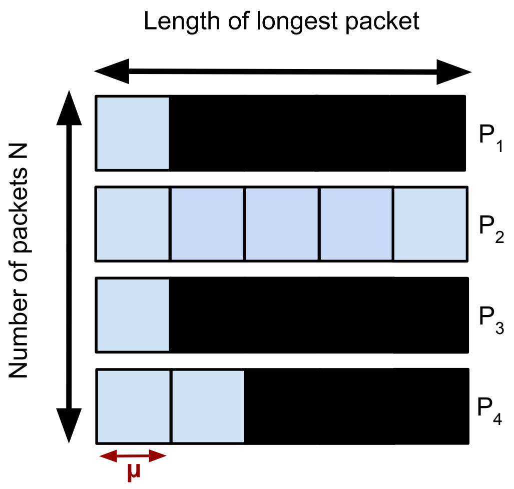


Figure 4.1: An example of a generation of $N = 4$ source packets with initial sizes μ, 5μ, 1μ, and 2μ, whereby μ denotes a prescribed packet size unit, e.g., $\mu = 10$ bytes. The packets with sizes 1μ and 2μ are padded out to the maximum packet size of 5μ before RLNC coding. The resulting padding overhead (represented by the black rectangles) is $O_G = 11/9 = 122\%$, i.e., the amount of padding overhead is larger than the amount of source data [4].

4.1.2 Pre-Processing for Generation Based RLNC

When forming a generation as displayed in Figure 4.1, two crucial elements should be taken into consideration [168], namely, the number of packets N to be mixed (referred to as the generation size) and the packet sizes, which can be a network constraint, e.g., taking into consideration that the MTU for Ethernet is 1500 bytes. Therefore, the generation should contain the maximum number of packets that do not introduce neither extra computational complexity nor decoding delays.

Without loss of generality, we consider the design and the transmission of one generation only in a point-to-point network. Let G be a generation formed of N source packets, $G = \{P_1, P_2, ..., P_N\}$ with corresponding sizes $L = \{L_1, L_2, ..., L_N\}$, and we denote the largest packet size in a generation as L_{max}, i.e., $L_{max} = \max_{1 \leqslant i \leqslant N} L_i$. The pre-processing to move from the packet level to the macro-symbol level results in having N packets with corresponding sizes $\Lambda_1, \Lambda_2, \cdots, \Lambda_N$ in macro-symbols.

Furthermore, we define the degree distribution as the number of macro-symbols chosen altogether to generate the coded macro-symbol number λ of the coded packet C_i:

$$\Delta_\lambda = \begin{cases} N & \text{if } \lambda = 1 \\ N - \sum\limits_{\ell=1}^{\lambda-1} \Pi_\ell & \text{if } \lambda \in \{2, \cdots, \Lambda_{\max}\} \end{cases} \tag{4.2}$$

where $\Lambda_{\max}$ is the total count of packets of size j, and Π_ℓ is the source MS degree of column λ, i.e., # of MSs in column λ position $= N - \sum_{\ell=1}^{\lambda-1} \Pi_\ell$.

4.1.3 Macro-Symbol Size μ Evaluation

As previously explained, the design of the macro-symbol could induce a small padding overhead inside the last macro-symbol of a packet, defined by $O_{MS}^{(i)}$ as follows:

$$O_{MS}^{(i)} = L_i \mod \mu. \tag{4.3}$$

For instance, if we choose μ to be equal to the symbol size (e.g., a byte when GF(2^8) is used), then $O_{MS}^{(i)}$ equals zero.

The overall padding overhead that is expected when moving from the packet to macro-symbol level in generation-based RLNC is [31]:

$$O_{MS} = \left(\mu\Lambda_{\max} - \sum_{i=1}^{N} O_{MS}^{(i)}\right) \Bigg/ \sum_{i=1}^{N} L_i. \tag{4.4}$$

It is important to study the effect of macro-symbol sizes on the zero-padded data that is processed with RLNC. Computing on the symbol level seems to be the accurate solution to seize the trailing zero symbols. However, a large portion of internet packets has the same size as the MTU, making the decoding process slower and bringing delay issues, which we discuss later in this chapter. Considering these concatenated blocks with a reasonable size is more convenient as it provides an acceptable trade-off between computational complexity and the ability to address zero-padding. Intuitively, using large MSs comes at the cost of hiding the padded zeros inside the last payload MSs. Thus, we shall study the effects of macro-symbol size on the overhead and the overall performance. We propose the use of the IP packets and video traces introduced in Chapter 3.

IP packets We run tests in order to estimate the average ratio of the padding overhead sent when tuning the macro-symbol sizes. We used a TCP traffic of 10^6 packets from a CAIDA trace [172] and formed generations based on the unidirectional flow of packets.

We set N_{max} as the maximum number of packets that could be combined in a generation, as described in Chapter 3.

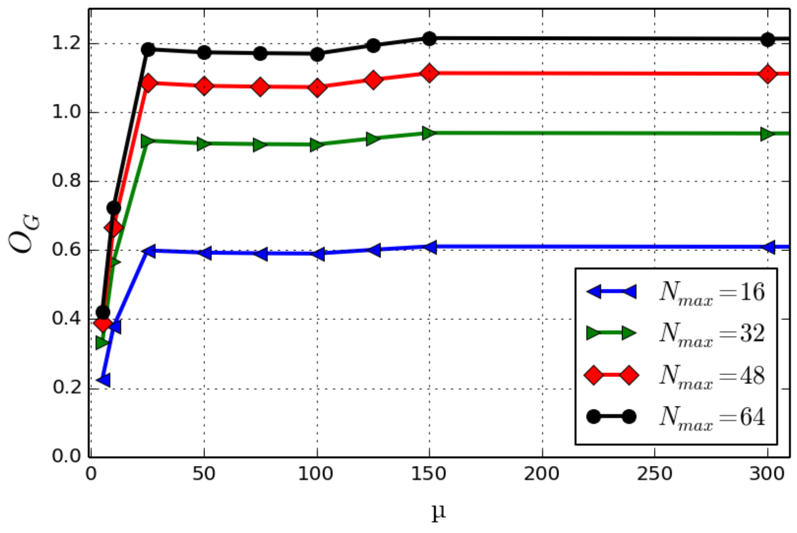
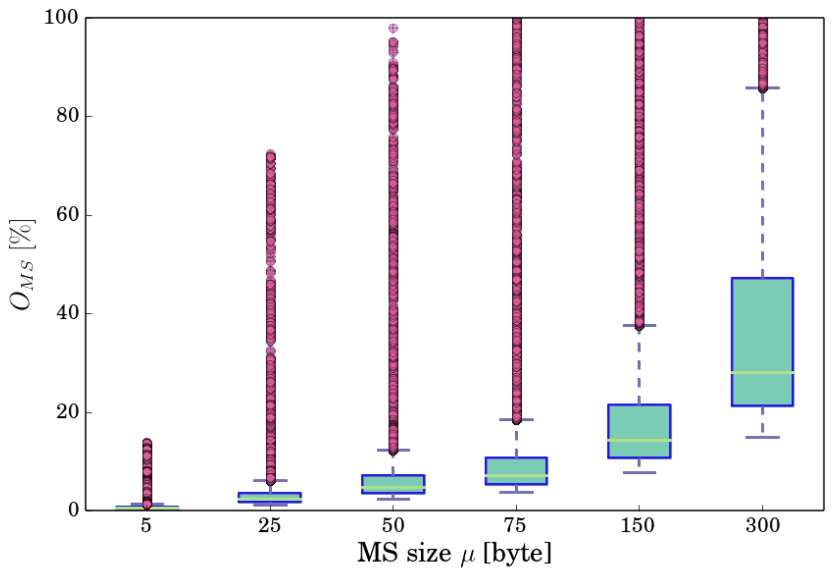


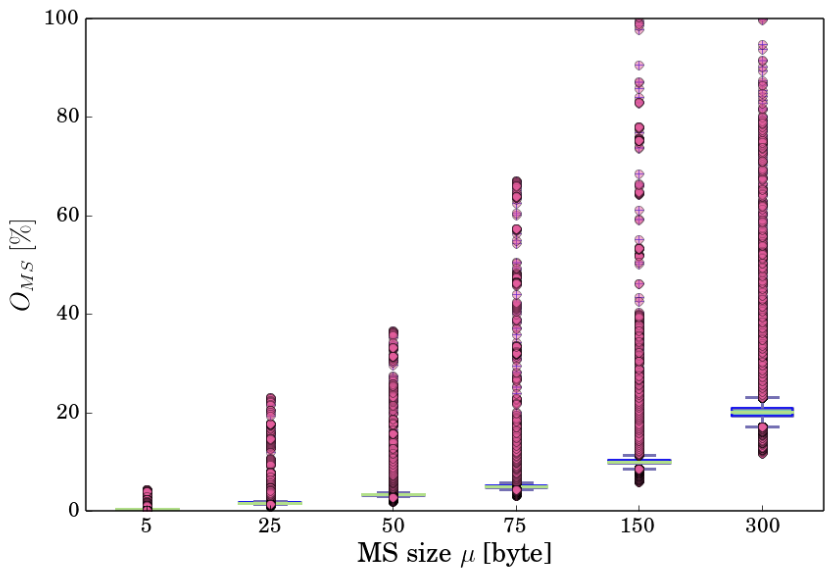
Figure 4.2: Ratio of zero-padded macro-symbols carried inside one coded packet in the CAIDA TCP traces [5].

Figure 4.2 illustrates the ratio of padding overhead included in a macro-symbol based generation on average that is relative to the amount of data in the generation $\sum_{n=1}^{N} L_n$, as a function of the macro-symbol size μ, for TCP flows from an online CAIDA trace. We observe that for small macro-symbols sizes, little padding overhead is expected to be created. For instance, its ratio is between 0.2 and 0.47 for $\mu = 10$ and a generation size N_{max} varying between 16 and 64. Furthermore, the generation size matters as well when using MSs, e.g., for $\mu = 150$ the padding overhead is around 1.2 for $N_{max} = 64$ and is around 0.6 for $N_{max} = 16$. Therefore, it is also crucial to carefully choose the generation size to employ in macro-symbol-based RLNC. However, the padding overhead stabilizes at $\mu = 25$ and above, which allows for a set of options to be considered based on the application. For example, the ratio of padding overhead is around 0.6 for $N_{max} = 16$, whether the macro-symbol size is $\mu = 25$ or $\mu = 300$.

Video Packets Figure 4.3 shows boxplots of the resulting padding overhead O_{MS} due to performing macro-symbol based RLNC, as in Eq. (4.4) in percentage for a generation of size $N = 64$ and different macro-symbol sizes μ. For the VP9 (HD) video in Figure 4.3b, we observe that for small macro-symbol sizes μ, the overhead is mainly present as outliers. Particularly, for small MS sizes between $\mu = 5$ and $\mu = 25$ bytes, the median padding



(a) SVC (CIF)



(b) VP9 (HD)

Figure 4.3: Percentage of zero-padding overhead for MS RLNC with different macro-symbol sizes μ for generations of $N = 64$ packets [4].

overhead is close to zero and the few individual outliers stretch to only about 20%. Unlike the behavior with the TCP trace in Figure 4.2, the padding overhead with MS RLNC increases for the video traces with increasing macro-symbol size μ. For example, we notice from Figure 4.3a that for the SVC (CIF) video trace, the median padding overhead comes near 20% for a macro-symbol size of $\mu = 150$ bytes. Nevertheless, such median MS RLNC padding overheads are still completely low compared to the median padding overheads with the standard RLNC, which we remind was on the order of 100% for SVC (CIF), see Figure 3.9. As for VP9 (HD), the median standard RLNC padding overheads were around 20% according to Figure 3.7. Note that choosing the macro-symbol size μ to be equal to the MTU size would result in overheads that are as large as or greater than the overheads in Figure 3.9. Specifically, larger overheads for $\mu = 1500$ bytes appear for generations with $L_{\max} < 1500$ bytes which require only padding to $L_{\max}$ in standard RLNC. We overall conclude from Figure 4.3 that reducing the macro-symbol size μ considerably diminishes the MS RLNC padding overhead. Moreover, the reductions are dramatic for macro-symbol sizes μ that are below 100 bytes.

4.1.4 Simple Macro-Symbol RLNC Scheme

The idea of performing network coding on macro-symbol is motivated by the fact that such small entities allow the control of the data at a finer grain without the need for padding in general.

Our encoding approach is faithful to the conventional RLNC generation based technique, with the exception that our approach encodes at the granularity of individual macro-symbols on the set of packets. Recall that $s_{i\lambda}$, with $i \in \llbracket 1 \; ; \; N \rrbracket$ and $\lambda \in \llbracket 1 \; ; \; \Lambda_i \rrbracket$, denotes the macro-symbol of the source packet P_i (row position i) and MS position (column position) λ. We denote by $\alpha_{\eta i}$ the coding coefficient mapped to the source packet P_i when creating the coded packet C_η, $\eta \geqslant 1$. We denote by $c_{\eta\lambda}$ the coded MS in the coded packet C_η (row position η) and the MS position (column position) η. Following the exact procedure in generation based RLNC, $c_{\eta\lambda}$ is created by linearly combining all the source MSs $s_{i\lambda}$, $i \in \llbracket 1 \; ; \; N \rrbracket$, in the column position λ with the weights provided by the coding coefficients $\alpha_{\eta i}$. This means that every source MS in the column position λ is multiplied with the corresponding coding coefficient $\alpha_{\eta i}$ for its row i. The products are then summed over the entire column, as shown in Figure 4.4. A coded macro-symbol $c_{\eta\lambda}$ can be formally expressed as:

$$c_{\eta\lambda} = \sum_{n=1}^{N} \alpha_{\eta n} s_{n\lambda}, \quad \eta \geq 1, \; 1 \leq \lambda \leq \Lambda_n. \tag{4.5}$$

Note that the degree Δ_λ gives the number of source macro-symbols that are linearly combined to generate the coded MS $c_{\eta\lambda}$.

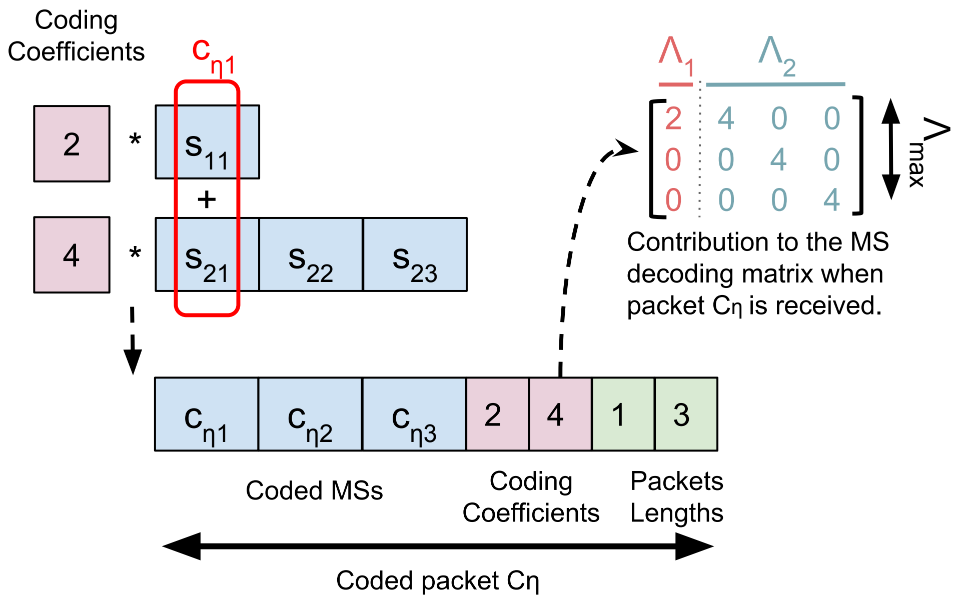


Figure 4.4: Example illustration of MS RLNC coding for a generation that consists of $N = 2$ source packets with maximum size $\Lambda_{\max} = 3$ MSs. The coded MS $c_{\eta\lambda}$ is created by linearly combining the source MSs $s_{i\lambda}$ in the column position λ with the random coding coefficients $\alpha_{\eta i}$. The coded packet C_η consists of the coded MSs $c_{\eta\lambda}$ from all the column positions λ, $\lambda = 1, 2, \ldots, \Lambda_{\max}$, and the coding coefficients $\alpha_{\eta i}$, $i = 1, 2, \ldots, N$; additionally, one coded packet of the generation has to carry to individual packet lengths Λ_i. The illustration on the right depicts the eventual matrix expansion of an encoding vector at the decoder [5].

As depicted in Figure 4.4, the coded packet C_η is constructed with the coded MSs $c_{\eta 1}, c_{\eta 2}, \ldots, c_{\eta \Lambda_{\max}}$ along with the coding coefficients $\alpha_{\eta 1}, \alpha_{\eta 2}, \ldots, \alpha_{\eta N}$. The coding coefficients require $N \log_2(q)$ bits. Additionally, one coded packet in the generation needs to convey the individual packet lengths $\Lambda_1, \Lambda_2, \ldots, \Lambda_N$ of the source packets involved in the considered generation to the destination to enable decoding. As a matter of fact, MS RLNC is not able to decode the generation exclusively from the coding coefficients as it is the case with the conventional RLNC. Recall that every packet is at most of $\Lambda_{\max}$ MSs long. Therefore, the overhead of sending the N packet sizes has an upper bound of $N \log_2(\Lambda_{\max})$ bits. It is not required to send the packet sequence numbers for MS RLNC decoding, just the packet lengths $\Lambda_1, \Lambda_2, \ldots, \Lambda_N$. In general, the coded packets carry a generation index when the data stream consists of several generations.

4.1.4.1 Coded Packet Size Distribution

As in conventional RLNC, the source needs to transmit as many coded packets $C_\eta,\ \eta = 1, 2, \ldots, K$, with $K \geq N$, as required to ensure the complete recovery of the data at the decoder. We consider a code rate $R = N/K = 1$ for ease of explanation. The MS RLNC encoding that we introduced enables a set of policies for setting the sizes of the encoded packets Φ_η in units of MSs, whereby one macro-symbol corresponds to one encoded macro-symbol $c_{\eta\lambda}$.

Full-Length Coded Packets A full-length coded packet policy generates N coded packets of size $\Phi_\eta = \Lambda_{\max}$ coded MSs each, with $\eta = [\![1\ ;\ N]\!]$. This encoding strategy corresponds to the standard packet granularity RLNC encoding, and it consequently incurs the padding overhead examined in Chapter 3. When effectively creating the coded macro-symbols following Eq. (4.5), the non-existent original symbols are considered to be zero-symbols, such as the four rightmost symbols s_{12}, s_{13}, s_{14}, and s_{15} in the source packet P_1, as displayed in the Figure 4.1. This means that all source packets are implicitly padded with zeros to the full-length of the largest packet size $\Lambda_{\max}$ of the generation.

Min-Sized Last Coded Packet The min-sized last coded packet policy generates $N-1$ coded packets that have the full-length of the maximum source packet size $\Lambda_{\max}$. The last coded packet's length is set to $\Phi_N = \Lambda_{\min} = \min\limits_{1 \leq i \leq N} \Lambda_i$, as shown in Figure 4.5. This allows the min-sized last coded packet policy to reduce the padding overhead by $\Lambda_{\max} - \Lambda_{\min}$ MSs compared to the aforementioned policy and the conventional RLNC.

4.1.4.2 General Macro-Symbol Decoding

We choose to perform MS RLNC decoding based on on-the-fly Gaussian elimination [62], due to its lower computational complexity compared to the standard Gaussian Elimination, which starts after receiving the entire batch of coded packets. Contrarily, the triangulation step starts as soon as a new coded packet C_η, i.e., a set of Φ_η coded MS is received. The MS RLNC decoding strategy consists of forming a triangular decoding matrix of dimension $\sum\limits_{i=1}^{N} \Lambda_i \times \sum\limits_{i=1}^{N} \Lambda_i$ by incorporating any received coded MS, starting from the very first one. The receiver decodes the original packets by exploiting the coding coefficients $\alpha_{\eta i}$ that are carried in the transmitted coded packets identically to conventional RLNC decoding.

The individual packet sizes $\Lambda_i,\ i \in [\![1\ ;\ N]\!]$, are crucial for decoding the original macro-symbols in MS RLNC. They determine the dimension of the decoding matrix resulting from every coded packet. Unlike standard RLN, every coding vector is expanded to a partial decoding matrix, as depicted on the right side of Figure 4.4, where $\Lambda_{\max}$ corresponds to the number of rows, and $\sum_{i=1}^{N} \Lambda_i$ to the number of columns, resulting in a matrix of size $\Lambda_{\max} \times \sum_{i=1}^{N} \Lambda_i$. Every column set of original symbols $s_{1\lambda}, s_{2\lambda}, \ldots, s_{N\lambda}$ can be completely recovered after receiving the coded macro-symbol in the column position λ of the coded packet number Δ_λ, i.e., after decoding exactly $\lambda + \Lambda_{\max}(\Delta_\lambda - 1)$ coded MSs, which is analogous to standard RLNC decoding [90, 171, 173].

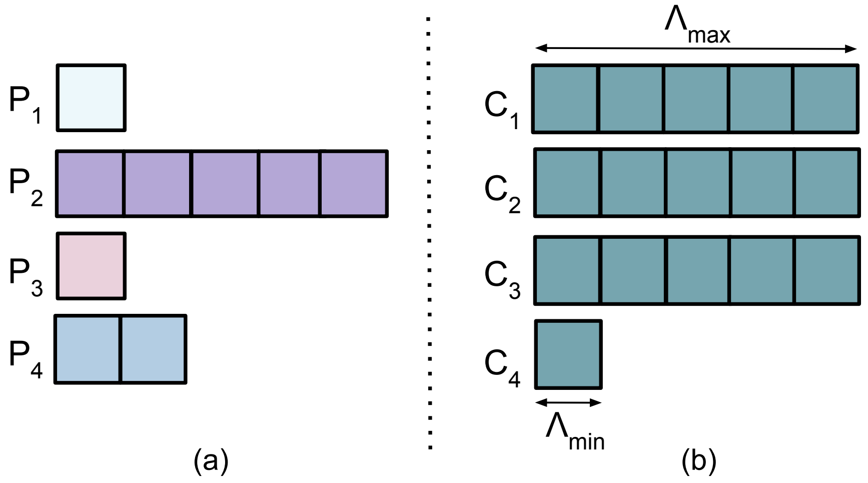


Figure 4.5: Example of the min-sized last coded packet policy; (a) a generation with $N = 4$ original packets, (b) coded packets after the encoding process: $N - 1 = 3$ coded packets have the full size of the largest original packet $\Lambda_{\max} = 5$, while the last has the size $\Lambda_{\min} = 1$ of the minimum length original packet [5].

4.1.4.3 Encoding Vector Overhead

Performing macro-symbol-based RLNC requires the communication of the packet sizes in MSs in addition to the random coefficients employed for encoding so that the receiver knows how to expand the coding coefficients to create the decoding matrix (see Figure 4.4). Nevertheless, the transmission of the sizes could be transmitted with the first coded packet only, if we have a perfect channel communication. In general, the encoding vector overhead is calculated in bits as:

$$\varepsilon_{MS} = (\log_2(q) + \log_2(N))N \quad (bits). \tag{4.6}$$

4.2 Progressive Shortening Scheme

We present the encoding and decoding process of the *progressive shortening* scheme, which is capable of generating unequal-sized coded packets. The encoding procedure is similar to the previously presented approach, with the exception of reducing the combination of macro-symbols that are expected to be decoded. Moreover, we show that the *progressive shortening* scheme has the full potential of keeping the conventional RLNC features, such as recoding, while ensuring that the padding overhead is not propagated as is the case with RLNC, thanks to its ability to heal a generation during the recoding phase.

4.2.1 Scheme Design

Here we discuss the encoding and decoding design for the progressive shortening scheme. We also explain the policies for generating unequal-sized coded packets.

4.2.1.1 Coded Packet Size Distribution

This scheme brings enhancements for both the sender and receiver compared to the conventional RLNC. With the *progressive shortening* schemes, coded packets are created with a size distribution that matches the length distribution Π_λ of the original packets, as shown in Figure 4.6. As a matter of fact, the number of original macro-symbols that are combined to form the coded MS $c_{\eta\lambda}$ in the column position λ is equal to the degree Δ_λ as in Eq. 4.2, i.e., the number of occurrences of the original macro-symbols in the column position λ. This means that the first coded macro-symbol $c_{\eta 1}$ (for $\lambda = 1$) in the coded packet C_η, with $\eta \leqslant K$, contains exactly N combined original macro-symbols. Moreover, the remaining coded macro-symbols, with column positions $\lambda \geqslant 2$, contain fewer combined original macro-symbols as controlled by the original packet length distribution Δ_λ.

If there are no corruption or loss of coded packets during the transmission, and if we suppose that there is no finite field dependencies [92, 174], then the receiver can decode the packets that have the length distribution Π_λ to recover the original packets with the same length distribution. Nevertheless, additional coded macro-symbols need to be transmitted in order to guard against packet corruption or loss, as well as linear dependencies. An elementary strategy would be to set a prescribed number of full-length coded packets Ψ beyond the number of full-length coded packets $\Pi_{\Lambda_{\max}}$ in the original packet length distribution before proceeding by shortening the coded packets according to the original packet size distributions. In Figure 4.6, the coded packet lengths are 5,5,1

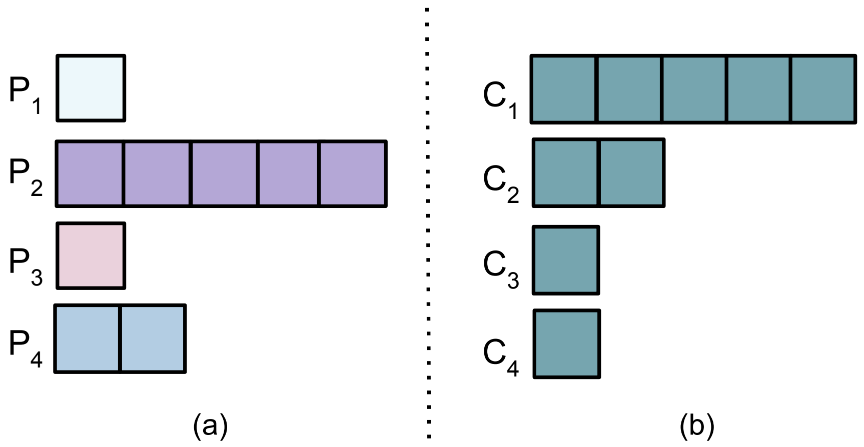


Figure 4.6: Best case scenario example of Scheme 2. (a) A generation of 4 original packets with unequal sizes. (b) Coding is performed sequentially on macro-symbols of different packets to create coded packets with progressively shorter sizes [5].

and 1 MSs for $\Psi = 1$. Another policy consists of shorten-and-hold, which holds the coded packet length at prescribed lengths. For example, one can send a certain number Ψ of full-length, as well as half-length coded packets before resuming the aforementioned shortening strategy. Specifically with Ψ, original packets of lengths 8, 7, 5, 5, 4, 2, 1, and 1 result in coded packet lengths 8, 8, 5, 5, 4, 4, 1, and 1. An alternative approach to control the number of coded macro-symbols is to exchange feedback between the sender and the receiver. Due to its simplicity, we adopt the elementary policy that adds Ψ coded of length $\Lambda_{\max}$ packets.

4.2.1.2 Shortened Coded Packet Decoding

We previously discussed the decoding with the min-sized last packet policy, which transmits coded packets of length $\Lambda_{\max}$, except for the last one that is of length $\Lambda_{\min}$. With the *progressive shortening*, the decoder receives coded packets that have a length distribution matching the length distribution of the original packets. For any of the progressive shortening policies, the decoding process starts with the very first coded packet received. Similar to the example on the right side of Figure 4.4, the encoding vector is expanded into a decoding matrix, and the conventional decoding of RLNC commences [61, 94, 175, 176].

It is crucial to check whether a coded packet is linearly dependent or not, as this determines whether to keep the packet for decoding or discarding it. The attractive feature of MS RLNC schemes is that they do not require the verification of the dependency within every single coded macro-symbol. Checking the first coded macro-symbol $c_{\eta 1}$ of a specifically coded packet C_η is sufficient to decide for the entire set of coded macro-symbols. Particularly, the same set of coding coefficients $\alpha_{\eta 1}, \cdots, \alpha_{\eta N}$ is used for the encoding of all macro-symbols present in coded packet C_η. This means that the random coefficients $\alpha_{\eta 1}, \cdots, \alpha_{\eta N}$ used for creating the coded macro-symbol $c_{\eta\lambda}$ is similarly used for $c_{\eta(\lambda-1)}$, if $\Delta_\lambda = \Delta_{\lambda-1}$, or a reduced set when $\Delta_\lambda \leq \Delta_{\lambda-1}$. As a result, the rank does not change, and the entire set of coded macro-symbols is discarded by the receiver. This reduces the computational effort invested in non-innovative packets.

4.2.2 Computational Complexity

We note that standard RLNC operates on N packets with size $L_{\max}$. It has an encoding complexity of $\mathcal{O}(L_{\max}N^2)$ and a decoding complexity of $\mathcal{O}(N^3)$ in the finite field [92, 175, 177–179]. Furthermore, we note for the state-of-the-art schemes that the simple bundling can be solved with a complexity $\mathcal{O}(N^2 + L_{\max}N_{SB}^2)$ when the *first fit decreasing* heuristic is used [170], which results in N_{SB}, with $N_{SB} \leqslant N$, packets of size $L_{\max}$ for the encoder, and a decoding complexity of $\mathcal{O}(N_{SB}^3)$. The encoding and decoding operations are conducted on a symbol by symbol basis when the Kodo library is used [171].

4.2.2.1 Macro-Symbol RLNC Encoding

The MS RLNC encoding complexity is proportional to the number of original macro-symbols considered for the generation's encoding. Particularly computing a coded packet C_η that consists of Φ_η coded macro-symbols requires exactly the computation of Φ_η macro-symbols. In particular, computing the coded macro-symbol $c_{\eta\lambda}$ that has a degree Δ_λ, requires that the Δ_λ macro-symbols involved in the encoding to be multiplied with the coding coefficient that corresponds to the packet they belong to. We note that each of these macro-symbols is of size μ in the finite field employed, and are at column position λ. Since the encoding is performed symbol-by-symbol in the corresponding finite field (e.g., byte-by-byte), going through all the symbols for processing and encoding is not an option, whether we perform the conventional RLNC or the MS RLNC schemes. As a result, the encoding of the Δ_λ macro-symbols available at the column λ that requires multiplications and summations, incurs a computational complexity $\mathcal{O}(\mu\Delta_\lambda)$ for the λ coded macro-symbol. The coded packet resulting from summing over all the considered coded macro-symbols λ, $\lambda \in [\![1\ ;\ \Phi_\eta]\!]$ and $\Phi_\eta \leqslant \Lambda_{\max}$, is obtained with a computational complexity

$$\mathcal{O}\Big(\mu \sum_{\lambda=1}^{\Phi_\eta} \Delta_\lambda\Big). \tag{4.7}$$

Therefore, the resulting computational complexity for encoding an entire generation that contains N original packets using MS RLNC is

$$\mathcal{O}\Big(\mu \sum_{\eta=1}^{N} \sum_{\lambda=1}^{\Phi_\eta} \Delta_\lambda\Big). \tag{4.8}$$

In comparison to the conventional RLNC encoding of a generation consisting of N original packets (all padded to the full-length of the largest packet $L_{\max}$), it follows that

$$\mu \sum_{\eta=1}^{N} \sum_{\lambda=1}^{\Phi_\eta} \Delta_\lambda \leqslant L_{\max} N^2. \tag{4.9}$$

We conclude that MS RLNC, that involves shortened coded packets, i.e., $\Phi_\eta \leqslant \Lambda_{\max}$, incurs a similar or lower computational complexity compared to the conventional RLNC.

We note that this analysis considers a code rate N/K equals to one, and no redundant packets are involved. An extra coded packet would simply incur the encoding complexity in Eq. 4.7 for MS RLNC, and $\mathcal{O}(L_{\max}N)$ for the standard RLNC.

4.2.2.2 Progressive Shortening's Sub-Decoders

When MS RLNC schemes are used, every encoding vector within a coded packet C_η is actually expanded into an $\Lambda_{\max} \times \sum_{i=1}^{N} \Lambda_i$ decoding matrix. Despite the fact that the coding coefficients are chosen uniformly at random from the finite field, their location and significance in the related decoding matrix are deterministic. Stacking the decoding matrices obtained after expanding every encoding vector results in a huge matrix with a dimension $N\Lambda_{\max} \times \sum_{i=1}^{N} \Lambda_i$, if we consider a code rate of one. Unfortunately, this decoding strategy undermines the decoding performance and is expected to incur higher decoding complexities that will curb the throughput and the overall performance of the MS RLNC approaches.

Practically, we can partition the decoding into sub-decoding steps, similar to the non-overlapping generations decoding in conventional RLNC [81]. Figure 4.7 displays an example of a generation decoding using sub-decoders. Each row corresponding to

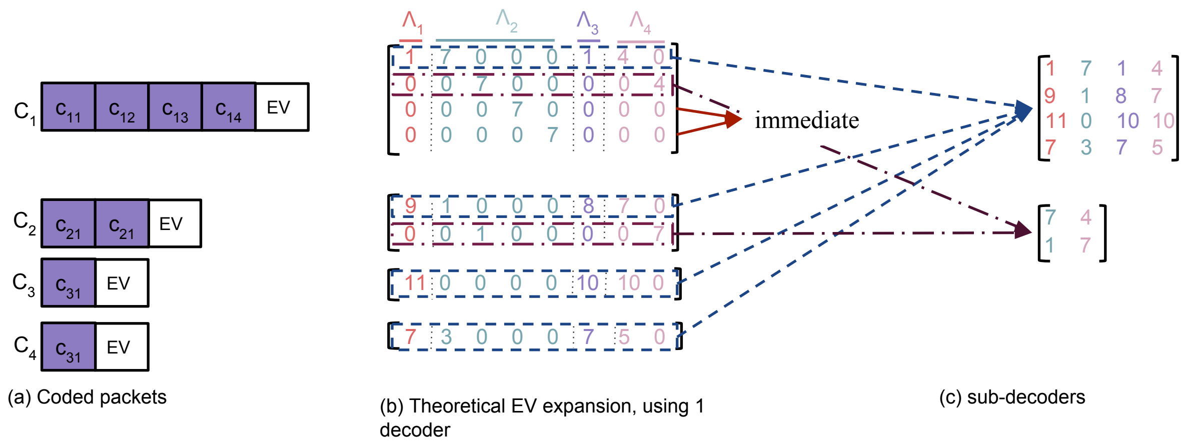


Figure 4.7: Decoding of unequal-size coded packets using sub-decoders for a generation of size $N = 4$; (a) a set of coded packets to decode. (b) The coding coefficients (EV) are expanded into a decoding matrix, where each row corresponds to the size of a coded packet in MSs. (c) The resulting sub-matrices are shaped based on the number of encoding coefficients involved in a row [6].

the encoding of a macro-symbol λ is merged into a specific sub-decoding matrix based on their degree Δ_λ. As such, any coded macro-symbol of a degree equaling one is immediately decoded, and further processing is omitted, e.g., c_{13} and c_{14} in Figure 4.7. We effectively integrated the sub-decoders in the Kodo library for performance evaluation of the progressive shortening scheme.

4.2.2.3 Progressive Shortening Decoding

We note that the decoding complexity of MS RLNC schemes, in general, follows the same principle, with the exception that the coded macro-symbols generated with the progressive shortening scheme follow the length distribution of the original packets.

It is clear that the decoding task when macro-symbols are used is the recovery of exactly $\sum_{i=1}^{N} \Lambda_i$ original macro-symbols. This is equivalent to decoding a matrix of dimension $\sum_{i=1}^{N} \Lambda_i \times \sum_{i=1}^{N} \Lambda_i$. However, as we previously explained, we rather decode the sub-matrices due to their expected reduced complexity. Nevertheless, inverting the progressive shortening sub-matrices using the Gaussian elimination yields a decoding complexity of [4]:

$$\mathcal{O}\Big(\sum_{\lambda=1}^{\Lambda_{\max}} \Delta_\lambda^3\Big) = \mathcal{O}\Big(\Lambda_{\max} N^3\Big), \tag{4.10}$$

which remains larger than the $\mathcal{O}(N^3)$ computational complexity of the standard RLNC, because [31]:

$$N^3 \leqslant \sum_{\lambda=1}^{\Lambda_{\max}} \Delta_\lambda^3 \leqslant \Big(\sum_{i=1}^{N} \Lambda_i\Big)^3. \tag{4.11}$$

We note that our analysis considers that the coded macro-symbols (on the columns $\lambda \in [\![1 \ ; \ \Lambda_{\max}]\!]$) are decoded independently, which is wasteful of the inter-knowledge between the columns and is also impractical. For an efficient implementation of the decoder, the columns must be decoded jointly as the same set of random coding coefficients are used for all the columns.

$$\mathcal{O}\Big(N^3 + (\Lambda_{\max} - 1)N^2\Big) = \mathcal{O}\Big(N^3\Big) \tag{4.12}$$

The decoding strategy that was previously proposed for this scheme is the same as of the traditional RLNC with the exception of the formation of a $\sum\limits_{i=1}^{N} \Lambda_i \times \sum\limits_{i=1}^{N} \Lambda_i$ triangular matrix. Since traditional decoding undermines the performance of the decoder due to the large dimensionality of the decoding matrix, we propose to perform decoding using *sub-decoders*. As it is depicted in Figure 4.7, each new coded macro-symbol is merged into multiple decoding matrices based on their degree. Any symbol having a degree of $\Delta_i = 1$ is immediately decoded and is omitted from further processing (see c_{13} and c_{14} in Figure 4.7.a). The computational complexity of this sub-decoder based decoding is $\mathcal{O}(\Delta_{max}^3) - \mathcal{O}(\Lambda_{max}\Delta_{max}^3)$, which is faster than the previously proposed decoding algorithm in [30].

Since the same coding coefficient is used for the encoding of all the macro-symbols corresponding to a given packet, this scheme has the favorable advantage of discarding an entire coded packet if its first coded macro-symbol turns out to be non-innovative. This is done by checking for linear dependency between the available decoding matrix and the new packet only with one of the sub-decoders. A symbol, therefore, is considered non-innovative if it does not increase the rank of a single decoder, in which case the processing of the remaining macro-symbols can be omitted.

4.2.3 Performance Evaluation

We implemented the introduced MS RLNC coding schemes using the Kodo library [171]. Building on the functionality of the Kodo library, we designed the MS RLNC encoding and decoding modules as wrappers of the corresponding Kodo modules. Our encoder wrapper maps the source data packets to source MSs. To encode the data, we generate random coefficients from the corresponding coding vectors and execute the Kodo encoder to produce the coded packets. Finally, we append the meta-data to the coded packet, i.e., the coding coefficients, the packet lengths (and the packet shifts for random shifting). For decoding, we map the coding vector from each received coded packet to a coding matrix. We then feed the coding matrix along with the coded data to the Kodo decoder [4].

We provide numerical results related to the *min-sized last coded packet* and *progressive shortening* schemes in terms of the data transmitted before total recovery for different finite fields. We consider performance comparison with data transmitted without coding, where we do not perform any processing and send every packet as it is. This approach is related to traditional routing, which does not guarantee the achievability of the network capacity. In contrast, network coding achieves network capacity in a variety of scenarios. As part of our evaluation, the "RLNC packet-level scheme" is also used for comparison as it represents a common technique that uses zero-padding in the case of heterogeneous packet sizes (see Figure 4.1). Since it operates on a packet-level instead of an MS level, zero-padding is treated as payload, which results in the network carrying useless padded data that obviously limits the benefits from RLNC.

4.2.3.1 Data Transmitted

In the following, we consider the evaluation of the total payload transmitted for different schemes including the ones introduced in this chapter, using the example in Figure 4.1 and H.265 (4K) video data frames from the online library of Arizona State University [13]. We note that "Prog. Sho." refers to the *progressive shortening*, "Min. Size" refers to the *Min-sized last coded packet*, "Theo. bou." refers to the theoretical bound obtained when the linear dependencies due to the field size are discarded, "RLNC" refers to the conventional RLNC, and "No coding" refers to the case where packets are simply being forwarded without coding (e.g., traditional routing that does not achieve the network capacity in general).

Basic schemes comparison We focus in this part on the decoding behavior and the payload overhead (excess MS) of the various aforementioned schemes through the previous example presented in Fig. 4.1. We run 10^4 encoding/decoding for each scheme, assuming that there are no erasures and that 0 encoding vector is acceptable unlike practical RLNC for an overall understanding.

Schemes	$q=2$	$q=2^4$	$q=2^8$	$q=2^16$
Min. Size	8.206	6.133	6.007	6
Prog. Sho.	8.18	6.13	3.007	0
RLNC	11.6	10.069	10.004	10
No coding	0	0	0	0

Table 4.1: Average extra number of coded MS needed to decode a generation of $N = 4$ input packets $(\Lambda_1, \Lambda_2, \Lambda_3, \Lambda_4) = (2, 5, 1, 2)$ [5].

The results in Table 4.1 show that our strategies require significantly less MSs on average for decoding compared to the naive solution adopted for RLNC. For small generations and small finite fields, both macro-symbol schemes perform similarly, since the sender transmits enough full-sized redundant packets to ensure possible MS recovery before the process of the output packet shortening starts. For this example, only two extra full coded packets are needed for finite fields of size $q = 2$ and $q = 2^4$, whereas, one redundant packet suffices for $q = 2^8$.

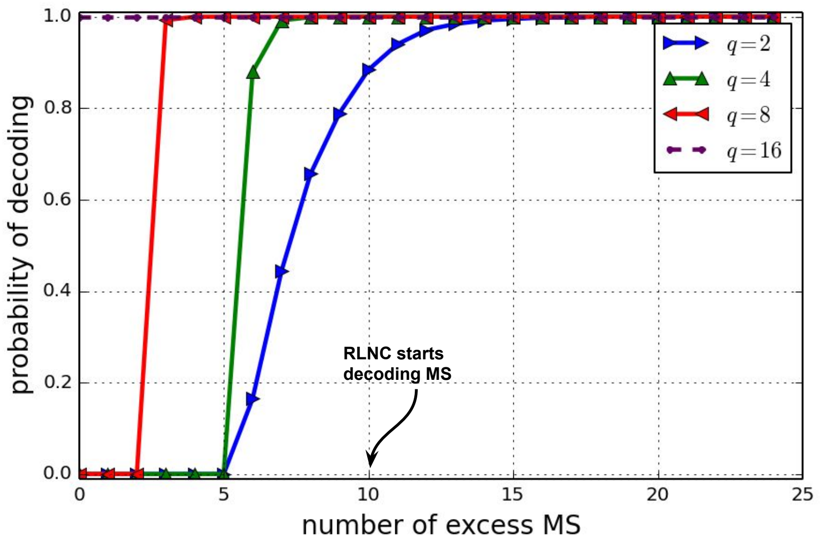
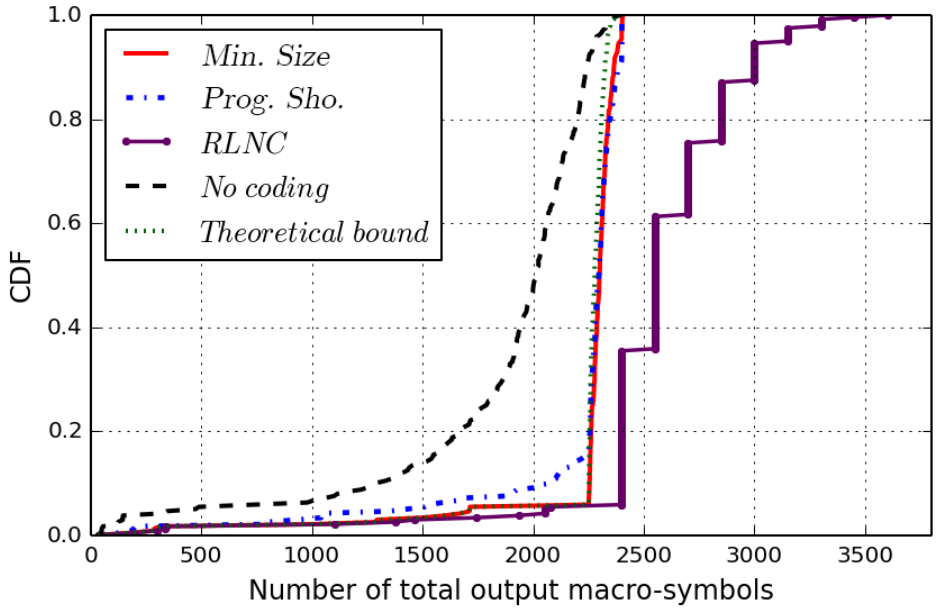
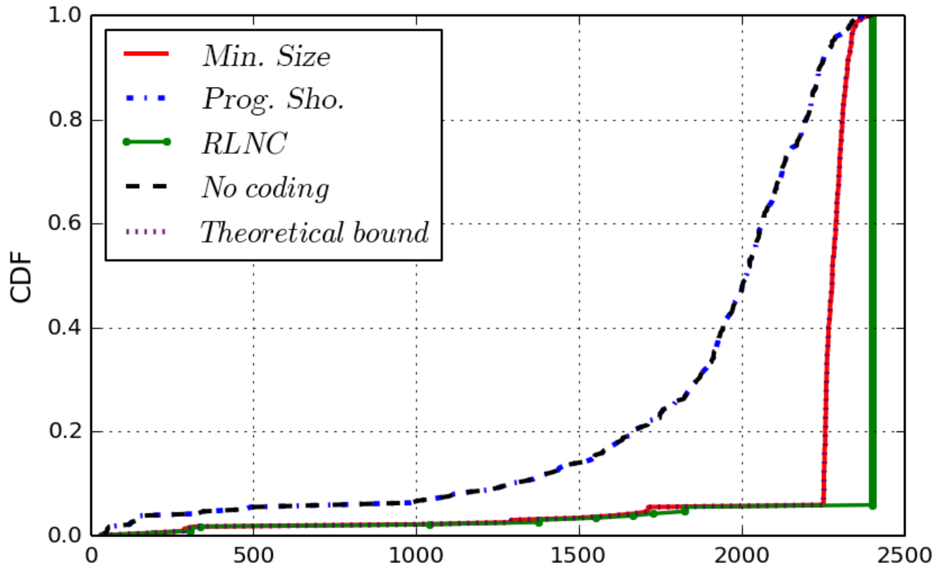


Figure 4.8: Probability of decoding in terms of excess output MS to decode a generation of $N = 4$ packets of lengths 2, 5, 1, 2 (the example in Figure 4.1) using *progressive shortening* scheme [5].

Even though redundant packets were needed when using some Galois Fields, Figure 4.8 shows that decoding using "Prog. Sho." can be finished with a probability close to one even before RLNC packet recovery starts. Without neglecting the fact that minimum or no prior payload overhead is created and transmitted, and the decoding cost in terms of the size of the data processed for decoding, such an approach outperforms the RLNC schemes when managing unequally-sized packets.



(a) CDF for a finite field size set to $q = 2$



(b) CDF for a finite field size set to $q = 2^{16}$

Figure 4.9: CDF comparison of the number of coded MS used in decoding H.265 (4K) video generations in each scheme, where $N = 16$, and $\mu = 10$ [5].

Video Frames We employ the MS RLNC schemes on a portion of 1600 original frames of the H.265 (4K) video trace from Table 3.1. The fragmentation to the MTU size resulted in 9952 packets with 82% of the packets being equal to 1500 bytes, which leaves a maximum expected gain in terms of padding overhead to be around 18% [5].

Figure 4.9 illustrates the CDF of the number of macro-symbols needed for decoding the portion of the H.265 (4k) video trace used for the specific finite fields GF(2) and GF(2^{16}), when a set of coding schemes are adopted. The generation size for the schemes that require coding is set to $N = 16$, and the macro-symbol size is $\mu = 10$. We observe that the conventional RLNC performs the worst, as the number of macro-symbols needed before finishing decoding the entire portion of the trace is larger compared to the remaining schemes. This is due to the practice of padding in RLNC, where this trailing data is counted as payload. Additionally, the *min-sized last coded packet* policy is slightly better than RLNC because of the shorter coded packets sent at the end of the generation transmission with size $\Lambda_{\min} = \min\limits_{1\leqslant i\leqslant N} \Lambda_i$.

On the other hand, the impact of the field size choice is noticeable between Figure 4.9a and Figure 4.9b. For instance, the *progressive shortening* scheme is capable of recovering around 2000 macro-symbols with a probability around 0.42 for GF(2^{16}), whereas this probability drops to around 0.08 with the small finite field GF(2). This difference is legitimate as the progressive shortening scheme inherits most of the RLNC features and weaknesses, including its sensitivity to the used field size. As a matter of fact, the progressive shortening scheme compensates for the linear dependencies or losses in general by simply sending more full-length coded packets, which unfortunately degrades its performance to the level which approaches the performance obtained with conventional RLNC. For such a small macro-symbol size, the trailing zeros that need to be filled in the last macro-symbols of a packet are considered as outliers.

Furthermore, Figure 4.9b shows that the performance with the progressive shortening coincides with the “No coding” scheme. This means that in the absence of linear dependencies, and the padding overhead resulting from the macro-symbol design is minimal, etc., the proposed macro-symbol scheme not only generates unequal-sized coded packets that match the size of the source packets but also dramatically reduces the padding-overhead created with generation-based RLNC.

4.2.3.2 Throughput

For the evaluation, we keep using a real-life trace transmission obtained from CAIDA. The trace contains over $12 \cdot 10^3$ packets from unidirectional flows between a sender and a receiver. We propose to evaluate the *throughput* and the *payload delivery efficiency* using our implemented *progressive shortening* scheme in the Kodo library, which is known for providing conventional RLNC and all its features. We note that we only use GF(2^8), i.e., we manipulate bytes only, without loss of generality to other finite fields, including our implementation for GF(2).

In computer networks, the throughput is the rate at which packets are successfully transmitted on a specific channel. We believe that the throughput is a metric of complexity, which gives additional insight into our analytical evaluation of the encoding and decoding computational complexities. In particular, it evaluates the rate at which the sender produces network coded packets and the receiver decodes them. For instance, for a given generation $g_i \in G$, where G is the complete set of N size network coding generations of a transmission session [6]:

$$\theta(g_i) = N^{-1} \sum_{j=1}^{N} ||f(y_j)|| \cdot \Delta t_j^{-1} \tag{4.13}$$

is the throughput of g_i, where $y_j \in g_i$ is the input (coded) packet, and $f(y)$ is the network coding (decoding) function. $||f(y_j)||$, in turn, represents the length of a given coded packet y_j, and Δt_j is the elapsed time for the whole coding (decoding) process.

To account for the encoder producing enough coded packets in the presence of losses, Eq. 4.13 is modified as follows [6]:

$$\theta^{\varepsilon}(g_i) = \frac{1}{2}\left(\theta(g_i) + (K-N)^{-1} \sum_{j=N}^{(K-N)} ||f(y_j)|| \cdot \Delta t_j^{-1}\right), \tag{4.14}$$

with K being the total number of encoded packets that depends on the expected loss probability ε. The throughput of the entire system under ε loss probability is derived as [6]:

$$\Theta^{\varepsilon} = (2 \cdot |G|)^{-1} \sum_{i=1}^{|G|} \theta_e^{\varepsilon}(g_i) + \theta_d(g_i). \tag{4.15}$$

$\theta_e^{\varepsilon}(g_i)$ and $\theta_d(g_i)$ represent the encoding and decoding throughputs, respectively. $|G|$ is the sum of all generations considered. We note that we only consider the actual coding/decoding operations to ascertain the complexity. Both procedures of the encoding and decoding involve other steps, including choosing random coefficients, as well as packet sorting. Nevertheless, they could be subject to more optimization effort that would prevent obtaining an efficient evaluation of the coding mechanisms.

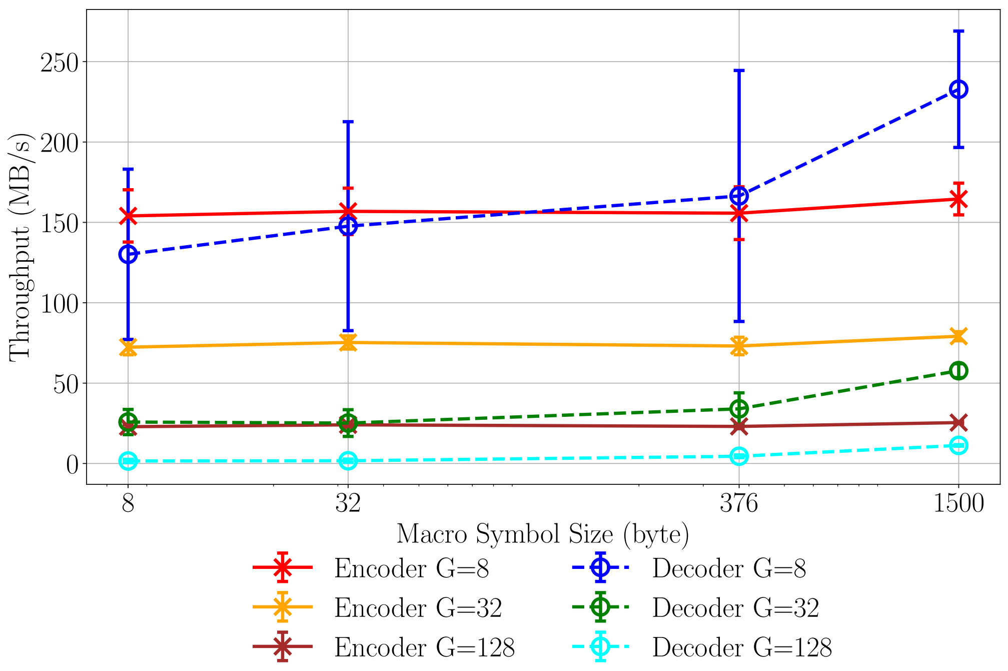


Figure 4.10: Encoding and decoding throughput for different macro-symbols μ.

"Because the throughput metric involves a time-based component, we dedicate a separate computer, specifically the *T470s ThinkPad* from Lenovo, with an *Intel i5-7200U* CPU of 2.5 GHz base frequency, 32–256–3072 Kb L1, L2, and L3 cache sizes respectively. We utilized a bare minimum installation of the *Linux Mint 18.3 64-bit* operating system, which is based on Ubuntu 16.04, and disabled the graphical shell and network functionality to ensure that the measurements have the same run-time conditions" [6].

Figure 4.10 shows the impact of the macro-symbol size μ , as well as the generation size on the throughput. We observe that the larger μ and the smaller N become, the higher the throughput gets, which can reach up to $\sim 170MB/s$ for $\mu = 376$ and $N = 8$. As for $\mu = 1500$, this is simply the case of using RLNC full-size packets, i.e., one packet is equivalent to one macro-symbol. Nevertheless, a larger μ does not solve the problem of the padding overhead meticulously. Consequently, we need to tune N and μ in a way that keeps a balanced tradeoff between zero-padding overhead and throughput (depending on the network requirements).

For closer insights, Figure 4.11 shows the average throughput of encoding and decoding for the standard RLNC and the *progressive shortening* as a function of the generation size N. As expected, the throughput is inversely proportional to the generation size (N), because the computational complexities are mainly led by the factor N. Moreover, we

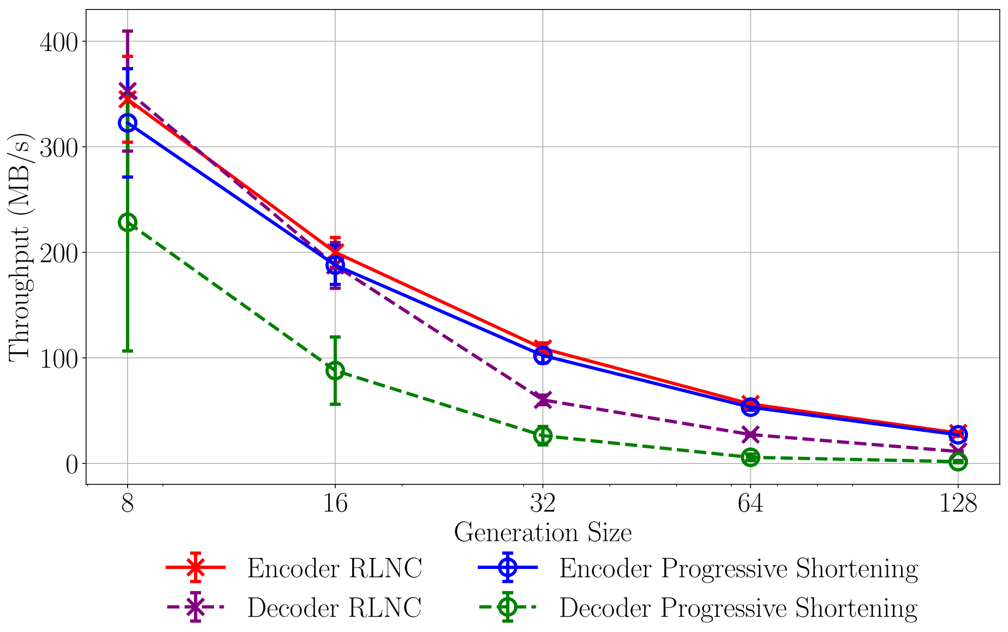


Figure 4.11: Encoding and decoding throughput of progressive shortening with $\mu = 32$, and conventional RLNC for different generation sizes [6].

notice a similar encoding performance for both schemes, but a lower decoding throughput for the *progressive shortening*, which is a good match with the results in [31]. Note that this scheme's implementation lacks a parallelization between macro-symbol decoders as they are agnostic to one another. The enhancement of this is a part of our ongoing efforts.

4.2.3.3 Payload Delivery Efficiency

We evaluate the *payload delivery efficiency* in order to confirm the gain of the shortened transmissions using the metric E that is defined as the ratio between the entire data received in bytes at the destination, without accounting for the errors and losses through the transmission, and the total data in bytes that was actually transmitted by the sender, including the redundant packets. E is expressed as:

$$E = \frac{Rx}{Tx}. \tag{4.16}$$

Table 4.2 showcases the payload efficiency advantage of using the *progressive shortening* scheme over conventional RLNC. We find that the scheme's efficiency is enhanced and clearly outperforms standard RLNC, referred to as the case with $\mu = 1500$, for all the generation sizes considered. As we anticipated, the largest gains are obtained when larger

Table 4.2: Payload delivery efficiency E of the progressive shortening scheme for different generation sizes N and macro-symbol sizes μ, in a loss-free point-to-point communication session.

μ \ N	8	16	32	64	128
8	95.97%	96.03%	96.04%	96.05%	96.06%
32	95.86%	95.88%	95.92%	95.92%	95.93%
376	94.15%	94.21%	94.24%	94.26%	94.27%
1500	88.61%	88.64%	88.65%	88.65%	88.66%

generation sizes and smaller macro-symbol sizes are employed. The macro-symbols have the finest control and granularity, and they adapt to even the smallest changes in packet lengths. We note that the gain in efficiency is contrasted by the complexity of solving different sub-decoders with high generation sizes, as we previously showed in Figure 4.11. Moreover, the progressive shortening scheme's efficiency reaches only 0.95 for $N = 128$ because we still carry a small portion of zero-padding overhead (see Eq. 4.4).

4.2.4 Recoding

The design of the progressive shortening scheme allows natural preservation of the salient features of standard RLNC, including *recoding*. As reviewed in Chapter 2, the recoding procedure in RLNC happens in a way that the coded payload and the newly drawn random coefficients are combined with the previous ones that are already provided by the coded packet to recode. On the other hand, we proved in Chapter 3 that the problem of the padding overhead is aggravated with RLNC when transmitting coded unequal-packets via a wireless mesh network, and showed that it increases linearly with the number of hops. This padding overhead, along with the losses on the links, will in turn cause a decline in throughput. In the remaining part of the chapter, we analytically characterize the expected padding overhead of the progressive shortening scheme and compare it through numerical results to the overhead generated with RLNC. We also note that this part was mainly published in [6].

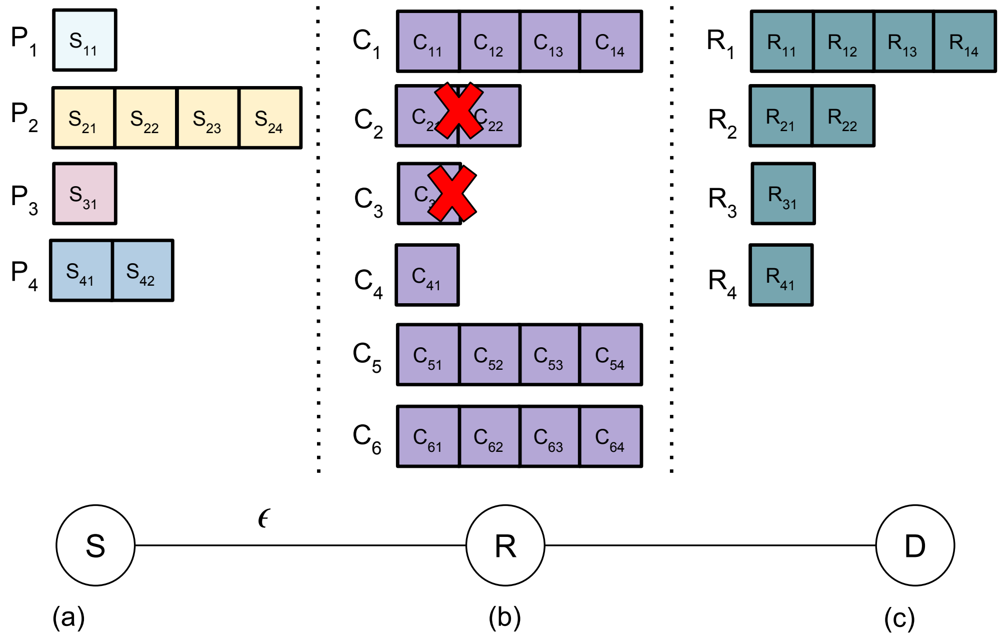
4.2.4.1 Characterization

It is a fact that progressive shortening needs to be accompanied by a policy that counters the occurring losses, corruptions, or linear dependencies among the coded packets. The simplest and most convenient way is to simply send extra full-length coded packets once the transmission with the progressive shortening scheme is done. As such, it wrongly appears that recoding the newly created batch of coded packets will result in changed length distribution of the recoded packets. However, as the original packet sizes are carried along with the coding coefficients on the coding vector, the task of determining the packet sizes resulting from recoding becomes trivial. Accordingly, the recoder checks within one of the coded packets (if multiple coded packets carry the sizes) in order to learn about the mechanism to adapt when generating new coded packets with sizes matching the lengths of the original ones [6]. This feature of generation healing during recoding is also performed with standard RLNC. The difference resides in the fact that standard RLNC is naively transmitting the padding overhead over and over while maintaining the generation size to a fixed size N. Under perfect conditions, and the *progressive shortening* can intervene in eliminating the padding overhead that it needs to carry during one transmission due to redundancy, aside from updating the generation size to N for the following recoding step.

Figure 4.12 illustrates that receiving coded packets with sizes not matching the numbers carried on the encoding vector does not prevent the recoder from generating coded packets following the length distribution of the original packets. Even in the presence of losses, the padding overhead is compensated for in the next transmissions. In this example, macro-symbols c_{13} and c_{14} are of degree $\Delta_3 = \Delta_4 = 1$, as they result from the multiplication of the original macro-symbols s_{23} and s_{24}, respectively. The recoder can therefore easily detect that c_{53}, c_{54}, c_{63}, and c_{64} are just redundant coded macro-symbols. With this large choice of coded macro-symbols for recoding, we simply opt for sticking with the first occurring coded macro-symbols at Δ_3 and Δ_4. Thus r_{14} and r_{15} are coded copies of c_{13} and c_{14} respectively.

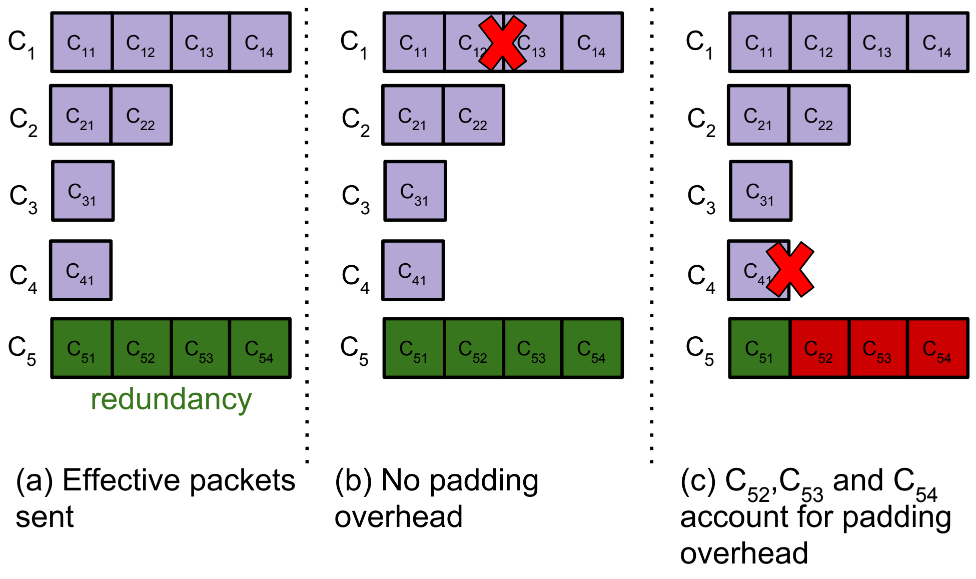
4.2.4.2 Expected Padding Overhead

It is important to note that the padding overhead is highly dependent on the size of the coded packets that are lost in the transmission. Due to the fact that we usually use the policy of compensation with full-length coded packets, a padding overhead could be generated in the case of smaller coded packet loss. The example in Figure 4.13 showcases



P_1 S_{11}
P_2 S_{21} S_{22} S_{23} S_{24}
P_3 S_{31}
P_4 S_{41} S_{42}
C_1 C_{11} C_{12} C_{13} C_{14}
C_2 C_{21} C_{22}
C_3 C_{31}
C_4 C_{41}
C_5 C_{51} C_{52} C_{53} C_{54}
C_6 C_{61} C_{62} C_{63} C_{64}
R_1 R_{11} R_{12} R_{13} R_{14}
R_2 R_{21} R_{22}
R_3 R_{31}
R_4 R_{41}
S
ϵ
R
D
(a)
(b)
(c)

Figure 4.12: Recoding of the shortening scheme in the case of losses; (a) a generation of $N = 4$ unequal packets, (b) two short coded packets are lost with a loss probability ϵ, but compensated with two full-length packets, (c) the recoder generates coded packets with length distribution matching the length distribution of the original ones in (a). [6].



C_1 C_{11} C_{12} C_{13} C_{14}
C_2 C_{21} C_{22}
C_3 C_{31}
C_4 C_{41}
C_5 C_{51} C_{52} C_{53} C_{54}
redundancy
(a) Effective packets sent
(b) No padding overhead
(c) C_{52},C_{53} and C_{54} account for padding overhead

Figure 4.13: Example of a coded generation where one random packet is lost; (a) the redundant packet matches the missing data and is not counted as overhead, (b) the redundant packet is larger and the difference is overhead [6].

two different situations with different coded packets being lost, where the size of the lost packet determines whether we are transmitting padding overhead or not. Ideally, as is the case in Figure 4.13.a, if the lost packet is a full-length packet, e.g., C_1, then the extra coded packet C_5 sent as compensation does not account for padding overhead. However, in Figure 4.13.b, 3/4 of C_5 size is padding overhead.

Nevertheless, the interesting capability of the progressive shortening scheme to heal a generation allows it to stop the propagation of this padding overhead, unlike what we previously showed analytically and through simulations for RLNC in Chapter 3. It is therefore appealing to characterize the expected padding overhead of a generation of N packets transported through an ℓ-hop network. We denote the expected overhead with one hop being considered by $\mathbb{E}[O]$, where $\chi_i \in [\![0 \; ; \; \sigma]\!]$ is the number of possible coded packets (among the first N transmissions, i.e., non-redundant packets) which might be lost due to the loss probability ε_i, with $\sigma \in [\![0 \; ; \; N]\!]$. This is expressed as follows [6]:

$$\mathbb{E}[O] = \sum_{\chi_i=0}^{N} \mathbb{E}[O|X=\chi_i] \cdot \mathbb{P}(X=\chi_i), \tag{4.17}$$

where $\chi_i = 0$ accounts for the case where no losses occur. $\mathbb{P}(X=\chi_i)$ is the probability of delivery of N packets, knowing that χ_i coded packets were lost, and defined by:

$$\mathbb{P}(X=\chi_i) = \binom{N}{\chi_i} \varepsilon_i^{\chi_i} (1-\varepsilon_i)^{N-\chi_i}. \tag{4.18}$$

We denote $\mathcal{J}^{(\chi_i)} = \{\mathcal{J}_1^{(\chi_i)}, \cdots, \mathcal{J}_r^{(\chi_i)}, \cdots, \mathcal{J}_{|\mathcal{J}^{(\chi_i)}|}^{(\chi_i)}\}$ as the set of indices of packets that could be lost due to the loss rate at the i^{th} hop, i.e., the tuples (here seen as sets) resulting in the combination of N packets among χ_i. Thus, $\mathcal{J}_1^{(\chi_i)} \neq \mathcal{J}_2^{(\chi_i)} \neq \cdots \neq \mathcal{J}_r^{(\chi_i)} \neq \cdots \neq \mathcal{J}_{|\mathcal{J}^{(\chi_i)}|}^{(\chi_i)}$, and $|\mathcal{J}^{(\chi_i)}| = Card(\mathcal{J}^{(x_i)}) = \binom{N}{\chi_i}$. As for $\mathcal{J}_r^{(\chi_i)}$, $r \in \{1, \cdots, |\mathcal{J}^{(\chi_i)}|\}$, it is a sub-set of $\mathcal{J}^{(\chi_i)}$, defined as:

$$\mathcal{J}_r^{(\chi_i)} = \{(\alpha_1, \alpha_2, \cdots, \alpha_{\chi_i}) : \alpha_j \geqslant \alpha_{j-1}, j \in \{2, \cdots, N\}\}. \tag{4.19}$$

Let $O(\mathcal{J}^{(\chi_i)})$ be the amount of the possible overhead (in macro-symbols), which is created due to loss rate ε_i, during one single transmission:

$$O(\mathcal{J}^{(\chi_i)}) = \sum_{n=1}^{|\mathcal{J}^{(\chi_i)}|} \left(\chi_i \Lambda_{max} - \sum_{a \in \mathcal{J}_n^{(\chi_i)}} \Lambda_a \right). \tag{4.20}$$

Therefore, the expected overall overhead, knowing that χ_i losses could occur is defined as:

$$\mathbb{E}[O|X=\chi_i] = O(\mathcal{J}^{(\chi_i)}) \cdot \frac{\chi_i}{|\mathcal{J}^{(\chi_i)}|}. \tag{4.21}$$

Consequently,

$$\begin{aligned}
\mathbb{E}[O] &= \sum_{\chi_i=0}^{N} \mathbb{E}[O|X=\chi_i] \cdot \mathbb{P}(X=\chi_i) && (4.22)\\
&= \sum_{\chi_i=0}^{N} \binom{N}{\chi_i} \varepsilon_i^{\chi_i} (1-\varepsilon_i)^{N-\chi_i} && (4.23)\\
&\cdot \left(\sum_{n=1}^{|\mathcal{J}^{(\chi_i)}|} \left(\chi_i \Lambda_{max} - \sum_{a \in \mathcal{J}_n^{(\chi_i)}} \Lambda_a \right) \right) \frac{1}{|\mathcal{J}^{(\chi_i)}|}. && (4.24)
\end{aligned}$$

For an ℓ-hop network, the expected overhead that is carried hop-by-hop until the destination is reached is defined by $\mathbb{E}[O^{\ell}]$ as follows:

$$\begin{aligned}
\mathbb{E}[O^{\ell}] &= \sum_{i=1}^{\ell} \sum_{\chi_i=0}^{N} \mathbb{E}[O|X=\chi_i] \cdot \mathbb{P}(X=\chi_i) && (4.25)\\
&= \sum_{i=1}^{\ell} \sum_{x_i=0}^{N} \binom{N}{x_i} \varepsilon_i^{\chi_i} (1-\varepsilon_i)^{N-\chi_i} && (4.26)\\
&\cdot \left(\sum_{n=1}^{|\mathcal{J}^{(\chi_i)}|} \left(\chi_i \Lambda_{max} - \sum_{a \in \mathcal{J}_n^{(\chi_i)}} \Lambda_a \right) \right) \frac{1}{|\mathcal{J}^{(\chi_i)}|}. && (4.27)
\end{aligned}$$

4.2.4.3 Numerical Results

Figure 4.14 depicts the size of the padding overhead, in megabyte (Mb), generated when using conventional RLNC and the progressive shortening scheme as a function of the number of hops (ℓ) within a network. As expected, the conventional RLNC's padding overhead is not only larger than the proposed scheme, but it increases linearly with the number of hops, whereas the padding overhead of the other scheme does not simply increase. We remind that we designed the progressive shortening to encounter the padding overhead, but by nature of its design it cannot always eliminate the overhead, and in this example, it is around 34 Mb. These results complement Figure 3.11 in showing that the progressive shortening scheme is capable of stopping the padding overhead from propagating when the network consists of more hops. For instance, the padding overhead reaches 68 Mb for $\ell = 8$ and 37 Mb for $\ell = 1$, but it remains almost unchanged for the progressive shortening scheme and is around 34 Mb for all $\ell \in [\![1 \ ; \ 8]\!]$.

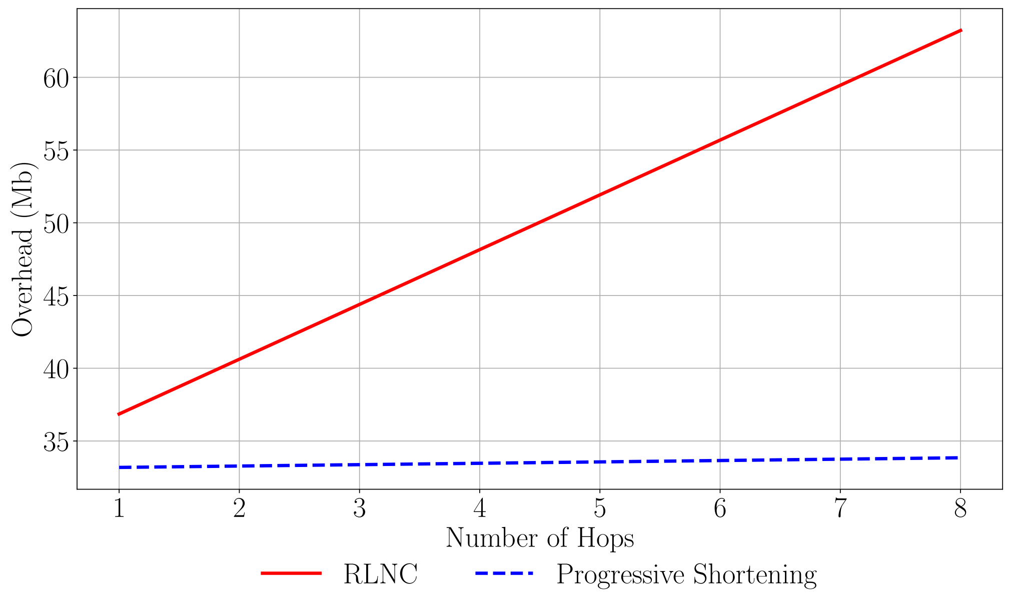


Figure 4.14: Overhead in ℓ-hops.

Based on the extensive statistical study in [3] using real-life TCP traces from [153] in a typical generation of size of $N = 50$ around 80 % of the transmitted data contains zero-padding overhead [3]. In a practical example, a *Variable Bit Rate* (*VBR*) video (the video trace employing the *CIF* video codec) would, for example, transmit more than 200 % overhead in extra (per generation) when coding is applied to it. The effect could even deteriorate more with large topologies, e.g., line networks, where the aforementioned overhead will be carried over and over within every single link, resulting in humongous bandwidth consumption.

Figure 4.15 depicts the percentage of maximum padding overhead generated due to the scheme design and the packet losses occurring due to channel imperfections. This percentage is obtained using the ratio of the transmitted overhead by the actual data to send per hop. At first glance, we expectedly notice that the RLNC percentage of overhead is ten times larger than that of the progressive shortening for a single lossy hop. It naturally follows that this percentage is held if the same losses, $\varepsilon_i = 0.4, \forall i \in \{1, \cdots, \ell\}$, occur at the next hops. As for the case where one link is lossy, e.g., $\varepsilon_1 = 0.4$, we observe that the RLNC overhead percentage decays fast (hop-by-hop) whilst it slightly drops from 8.3% to almost 2% for the progressive shortening. We conclude that the penalty of RLNC's padding overhead becomes more serious as the number of hops increases in the network. Interestingly, both schemes can heal a generation for the next hop, nevertheless, traditional RLNC is not equipped to compensate for overhead stemming from padding.

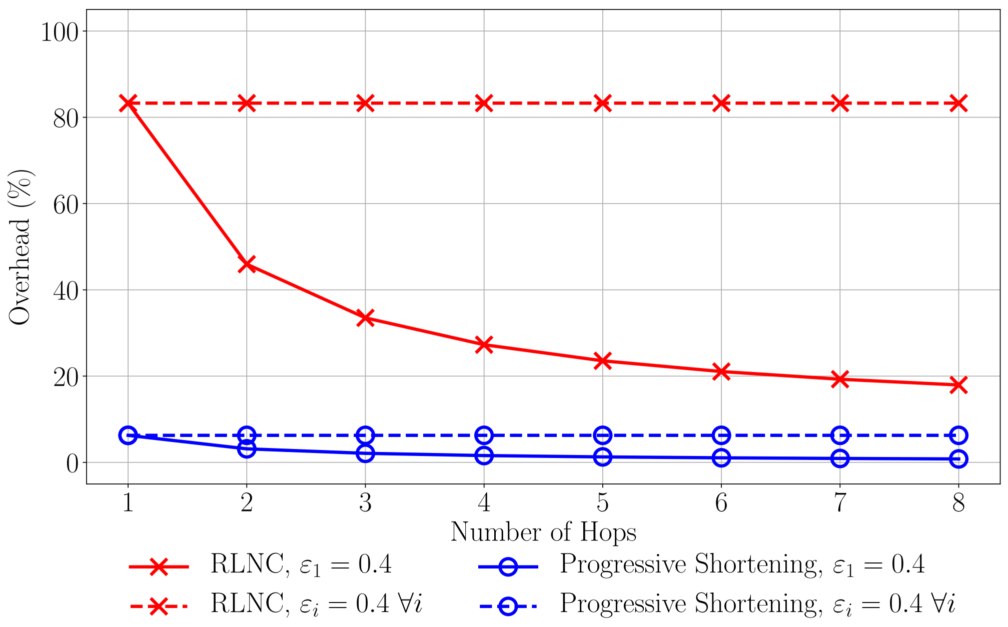


Figure 4.15: Percentage of worst case padding overhead in a network with ℓ-hops, where $\mu = 8$, $N = 32$, $\varepsilon_i = 0.4$, $\forall i \in [\![1\ ;\ 8]\!]$ [6].

4.2.5 Progressive Shortening and Robust Header Compression version 2 (RoHCv2)

In order to show the benefits of the *progressive shortening* scheme over the conventional RLNC, we propose to combine our scheme with a protocol header compression. Our goal is to further decrease the transmission overhead by applying the RoHCv2 [180]. According to [181], header compression techniques, as their name suggests, allow reducing the amount of data transmitted, mainly the packet header overhead, which results in energy savings especially for power-constrained devices such as sensors. In the following, we provide a general RoHCv2 overview along with the performance evaluation of the combination's efficiency.

4.2.5.1 Overview of RoHCv2

Header compression is a source coding technique that reduces the size of individual packet headers. The compression itself relies on the exploitation of header field progression characteristics between consecutive packets. As an example, IP addresses commonly stay constant during the transmission of a specific stream, and the RTP timestamp increases by a constant delta from packet to packet. Such characteristics can be utilized to reduce the size of the individual headers. It has been shown that Robust Header Compression can effectively guarantee 80-90 % header compression gain even on unreliable channels [182].

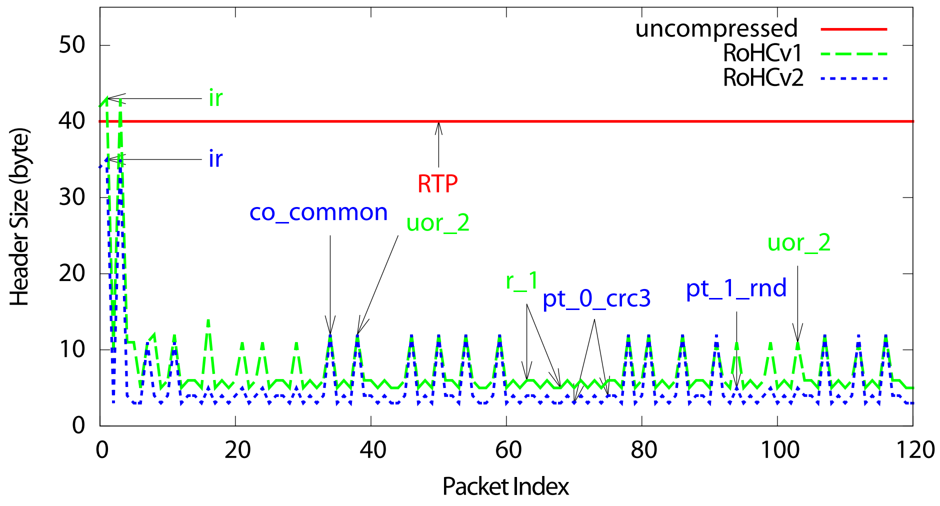


Figure 4.16: RoHCv1 [6].

However, the presence of channel losses, as well as multiple compressed streams, will inevitably lead to the generation of compressed headers with sizes ranging between 1 to 40 or more bytes (assuming RTP/UDP/IPv4 encapsulation). Even if the logical payload (data that is not compressed and is located after the headers) is expected to have a constant length over multiple packets, the presence of header compression would result in the transmission of compressed packets with various sizes. As an example, Figure 4.16 shows a comparison of packet header sizes and types generated during the compression of an RTP/UDP/IPv4 stream with Robust Header Compression version 1 and 2. In this case, even without the presence of packet losses, both compression standards produce compressed packets with sizes that differ either by a single byte or by at least five bytes between packets, even though the original header had a constant length. However, when the compressor is recovering from decompressor context desynchronization or preemptively generates redundant updates, this difference is further accented by even larger compressed packet headers.

4.2.5.2 Efficiency of the Joint Combination

In order to ascertain the efficiency of the joint combination of the progressive shortening scheme and the RoHCv2, we use header compressed streams based on UDP traces obtained from CAIDA. We assume for the operation of the RoHCv2, that the receiver delivers the decoded packets in order, and that sufficient redundant coded packets are sent within every generation, to assure the delivery to the decompressor. This enables the header compression, in turn, to function on a *unidirectional mode* with minimal requirements for robustness. As such, we guarantee to achieve the highest compression gain for header compression. As a matter of fact, we set the *optimistic repetition count* to 0 and the *unimode timeout* to 10^3. On the other hand, as the complete harmonization of header compression and RLNC is a part of our ongoing work, they both operate independently from one another, and the header compression operation is encapsulated by the network coding process.

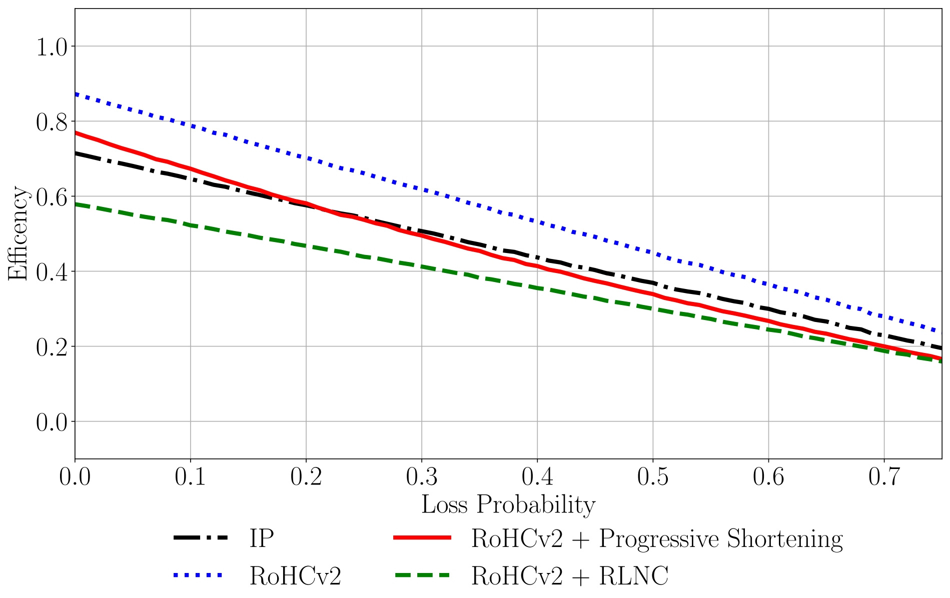


Figure 4.17: Network coding payload delivery efficiency for IP, RoHCv2, RLNC and progressive shortening for a UPD stream over simulated uncorrelated losses [6].

Figure 4.17 depicts the payload delivery efficiency as a function of the loss probability when different mechanisms are employed on a UDP stream with uncorrelated losses. We observe a gain for an artificial hard-to-compress RTP/UDP/IP stream, based on the features considered in [183]. RoHCv2 is about 20 % more efficient compared to the baseline IP. Additionally, conventional network coding when boosted with RoHCv2 still performs lower than the IP. Nevertheless, the combination of progressive shortening and RoHCv2 surprisingly produces better efficiency than the standard RLNC. This is explained by the considerable reduction of the padding overhead, which would normally hinder the gains from the header compression adoption. Moreover, the efficiency of RoHCv2 represents the maximum attainable gains. We note that dynamic configuration should be adopted when more robustness is required [183, 184].

4.3 Conclusions

This chapter was dedicated to the design of the progressive shortening scheme that aims to handle the heterogeneity of real data packets, as well as the padding overhead created with linear coding techniques, e.g., RLNC. This approach relies on macro-symbols, which are subsets of the packets allowing to drastically reduce the padding overhead compared to RLNC. We stress that progressive shortening has an exclusive and alluring ability in generating unequal-sized coded packets.

Furthermore, we promote progressive shortening as a fully-fledged network coding technique that inherits all the RLNC features, including recoding that enables intermediate nodes to be engaged and no longer passive entities. Our numerical results in a line network show that our scheme is capable of recovering a recoded generation without propagating the padding overhead, resulting from excess transmissions in lossy networks, unlike standard RLNC. Moreover, combining the progressive shortening scheme and the RoHCv2 yields a 20% higher payload delivery efficiency compared to the standard RLNC.

Despite the fact that the progressive shortening scheme fulfilled its task in reducing the padding overhead of real-life packets coding using conventional RLNC, it did not exploit these gaps on which it was operating to reduce the number of coded packets transmitted. In chapter 5, we introduce other macro-symbol-based RLNC schemes that reduce the number of coded packets needed compared to the original generation size whenever variable-sized packets are to be coded.

Chapter 5 | Shifting-Based Approaches

In this chapter, we advocate for novel approaches for macro-symbol-based RLNC that mainly rely on an RLNC compatible pre-coding stage, consisting of shifting the packets before performing macro-symbol encoding. The idea originated from the fact that the progressive shortening approach was not able to take advantage of the length distribution of the packets to encode in decreasing the number of coded packets it generates. We observed that the padding locations in shorter packets could be adequately exploited to effectively reduce the number of coded packets needed to decode a generation. Our random and deterministic shifting-based schemes allow the variation in the degree distribution of the macro-symbols corresponding to specific columns. The task is, therefore, to leverage the zero-padding-reduced feature of macro-symbol-based schemes, while reducing the decoding latency of the scheme. This means that each individual macro-symbol can be decoded using less coded packets. We analytically derive an exact decoding probability of our schemes and validate it using simulation results. Additionally, we make a practical implementation and perform an extensive evaluation using our benchmark video traces, including a multi-hop performance with intermediate nodes performing recoding operations and processing speed of the enhanced decoders. Furthermore, we provide a full performance comparison of all the RLNC related schemes that we are aware of for overcoming the padding overhead issue.

The second part of this chapter introduces another macro-symbol and shifting-based scheme, called *Random Shift and XOR* (RaSOR), which does not employ any coding coefficients, but simply performs XORing operations over the shifted macro-symbols for encoding and decoding. This approach is characterized by generating less coded packets compared to the original generation size for unequal-sized packets, as well as linear encoding and decoding complexities, which could promote macro-symbol coding for power-constrained devices. Finally, we note that the material for this chapter has been previously published, mainly in [4, 30].

5.1 Shifting-Based MS RLNC

The design of the *progressive shortening* forced the leftmost column of original macro-symbols to have a degree $\Delta_1 = N$, as shown in Figure 4.6, with a possibility that more columns have a degree N. Despite the flexibility that the macro-symbol concept gives for handling the encoding and decoding of variable-sized packets, this scheme's design poses a critical aspect for decoding, since at least N packets are required to recover these sets of original N combined macro-symbols. If the longest column of the original macro-symbol is reduced, i.e., the maximum degree defined by $\max\limits_{1 \leqslant \lambda \leqslant \Lambda_{\max}} \Delta_\lambda$, then the number of macro-symbols combined with one another is reduced, which leads to the reduction of the overall number of coded packets needed to decode a generation. Shifting the packets is a cheap way to rearrange them in a way that reduces the maximum degree of a generation. In the following, we present our shifting-based schemes and evaluate their performance analytically, as well as through simulations.

This section proposes two simple mechanisms that judiciously exploit the packet size heterogeneity to reduce the number of coded packets needed for decoding while maintaining low encoding and decoding costs.

5.1.1 Design of Shifting MS RLNC Schemes

We introduce the pre-coding step of shifting and the macro-symbol-based approaches for the variable-sized packets. As the shifting allows the redistribution of the degree of coded macro-symbols for encoding and decoding.

5.1.1.1 Random Shifting

This pre-coding moves the original packet P_i from its original position using a random offset to the right ϕ_i, drawn randomly from a uniform distribution over $[\![0\ ;\ \Lambda_{\max} - 1]\!]$. As such, the original macro-symbol $s_{i\lambda}$, in the column position λ of the original packet i is shifted to a higher column position, as depicted in Figure 5.1. The moving to a higher indexed column position wraps around at column position Λ_{max}, which means that the macro-symbol $s_{i\lambda}$ is moved to the column position $\lambda + \phi_n \mod \Lambda_{\max}$. The encoding process of this scheme is similar to the general MS RLNC approaches that we explained in Chapter 4, except for adding the shifts to the coded packet's header.

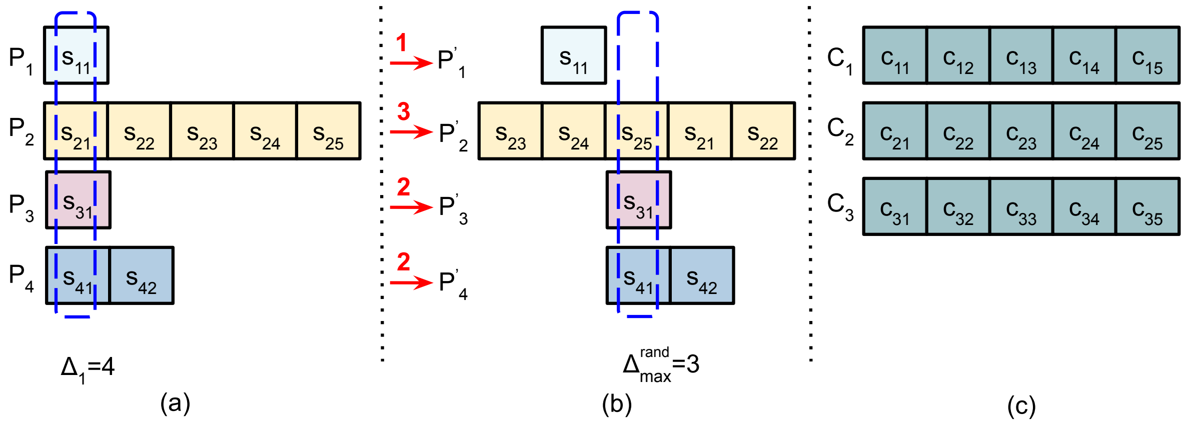


Figure 5.1: Example of random shifting: (a) generation of $N = 4$ original packets with unequal sizes; the maximum degree is $\max_{1 \leqslant \lambda \leqslant \Lambda_{\max}} \Delta_\lambda = \Delta_1 = 4$. (b) pre-coding: randomly shifted packets before MS RLNC encoding; the shifting reduced the maximum degree to $\max_{1 \leqslant \lambda \leqslant \Lambda_{\max}} \Delta_\lambda^{\text{rand}} = \Delta_{\max}^{\text{rand}} = 3$ [4].

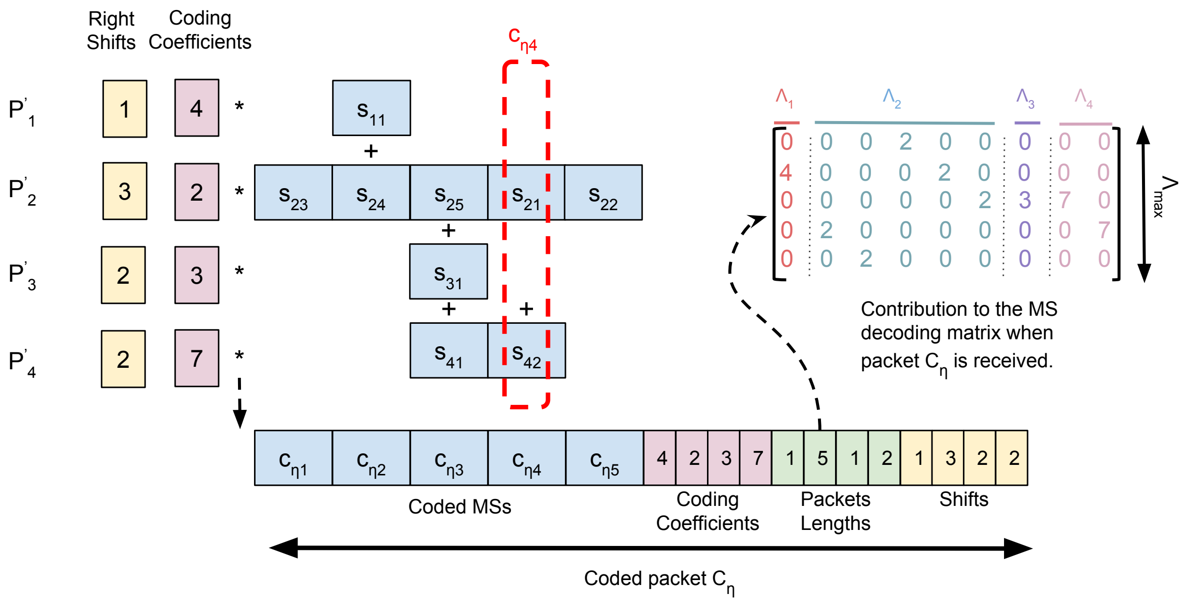


Figure 5.2: Example of MS RLNC encoding after random shifting for generation size $N = 4$ and maximum source packet size $\Lambda_{\max} = 5$ MSs with the eventual matrix expansion of the encoding vector [4].

As these shifts are generated only once per the transmission of a generation, they are communicated only once, similarly to the packet sizes. The encoding of the example in Figure 5.1 is detailed in Figure 5.2.

The drawback of the random choice of the shifts is that we might end up by having the maximum degree equal to the number of original packets N, $\max_{1 \leqslant \lambda \leqslant \Lambda_{\max}} \Delta_\lambda^{\text{rand}} = N$.

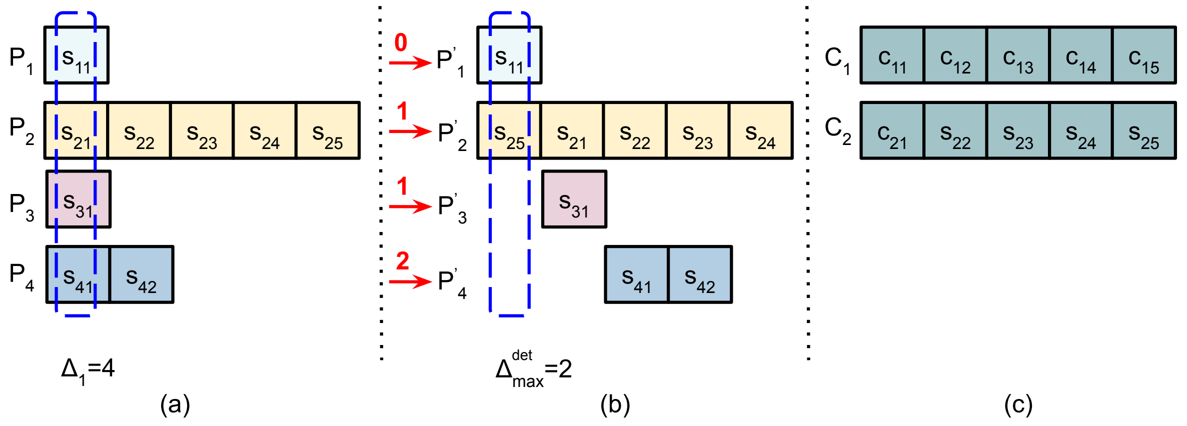


Figure 5.3: Deterministic shifting: (a) Generation of $N = 4$ source packets with unequal sizes. (b) The deterministic shifting has reduced the maximum MS source degree to $\Delta^{\mathrm{det}}_{\mathrm{max}} = 2$ [4].

On the other hand, this scheme could have the advantage of supporting privacy and security mechanisms in general. Altogether a deterministic approach to shifting definitely minimizes the maximum number of original macro-symbols to be encoded.

5.1.1.2 Deterministic Shifting

This deterministic mechanism is inspired by the *chain and fragmentation* scheme [169], where the original packets are consecutively chained to form a bulk data set, and then are fragmented into equal packets based on a fragmentation size choice F_s. As we discussed in Chapter 4 this approach mitigates the padding overhead but incurs substantial signaling overhead. In order to avoid both the bulk of padding overhead and the signaling overhead, we do neither fragment nor bundle the packets as we employ macro-symbols instead.

We deterministically right shift the original macro-symbols in a way that the successive original packets essentially form a long chain, which continuously cycles through the column positions $1, 2, \ldots, \Lambda_{\max}, 1, 2, \ldots \Lambda_{\max}, 1, 2, \ldots$. More specifically, we right shift the first MS $s_{i,\lambda}$ of source packet i into the column position following the position of the last MS $s_{i-1,\Lambda_{i-1}}$ of the preceding source packet $i-1$ *after* the right shifting of source packet $i-1$. Specifically, the first original packet P_1 ($i = 1$) is not shifted by nature, i.e., the last macro-symbol $s_{1\Lambda_1}$ of the first original packet stays in column position Λ_1 corresponding to the length (in MSs) of the packet $i = 1$. The first macro-symbol of the second packet P_2, i.e., macro-symbol s_{21} is right shifted into the position immediately to the right of column position Λ_1, i.e., into position $\Lambda_1 + 1 \mod \Lambda_{\max}$ (where the $\mod \Lambda_{\max}$ accounts for the wrap-around). Then, the other MSs of source packet $i = 2$ are placed successively to the right (with wrap-around, if needed) of its first macro-symbol, as illustrated in the second row of Figure 5.3. This right shifting continues for the subsequent source packets. In the example in Figure 5.3, the last source MS of packet $i = 2$ is shifted into column position 1; thus, the first MS s_{31} of packet $i = 3$ is right shifted into column position 2, and so on.

The chaining of the original packet through the right shifting effectively creates one long chain with $\sum_{i=1}^{N} \Lambda_n$ macro-symbols. This chain fills the column positions $1, 2, \ldots, \Lambda_{\max}$ completely for a total of $\lfloor \sum_{i=1}^{N} \Lambda_i / \Lambda_{\max} \rfloor$ times and "spills" over $(\sum_{i=1}^{N} \Lambda_i / \Lambda_{\max})$ mod $\Lambda_{\max}$ macro-symbols into the next row. Therefore, the maximum original macro-symbol degree is [4]

$$\Delta_{\max}^{\text{det}} = \left\lceil \sum_{i=1}^{N} \Lambda_i \Big/ \Lambda_{\max} \right\rceil . \tag{5.1}$$

This maximum original macro-symbol degree $\Delta_{\max}^{\text{det}}$ constitutes a lower bound for the expected number K of coded packets needed for decoding the generation of N source packets at the destination, i.e., $K \geqslant \Delta_{\max}^{\text{det}}$.

We note that the shifts do not need to be encapsulated into the header with the deterministic shifting, as their values are deduced from the packet sizes, which are carried in the packet header, as all the MS RLNC schemes. Both shifting schemes can adopt two transmission policies. The first consists of transmitting $K,\ K \geq \Delta_{\max}^{\text{rand/det}}$, equal size coded packets $\eta,\ \eta = 1, 2, \ldots, K$, each containing $\Phi_\eta = \Lambda_{\max}$ coded macro-symbols. The second follows the min-sized last coded packet policy as described in Chapter 4. It transmits $\Delta_{\max}^{\text{rand/det}} - 1$ full-length coded packets followed by a "spill-over" packet consisting of $(\sum_{i=1}^{N} \Lambda_i / \Lambda_{\max}) \mod \Lambda_{\max}$ macro-symbols. For simplicity, we opt for generating equal size coded packets following the first policy

5.1.1.3 Recoding

We note that the shifting of the source packets is a pre-coding step that occurs only at the encoder. Recoding in the network does not affect the properties and performance described from a receiver's perspective. Recoding is oblivious of the pre-coding shifting and will operate as in standard RLNC. However, as the shifting allows the reduction of the generation size in general, the recoding is expected to maintain this number. As the coded packets generated using the shifting schemes are full-size, it makes more sense to recode the generation on the packet level instead of on the macro-symbol level to guarantee a reduced computational complexity. Since RLNC, in general, is finite field-dependent, therefore whether we multiply a coding coefficient by a set of macro-symbols that form a packet or simply an entire packet, the result is the same as the multiplication operations are performed on a symbol by symbol basis (e.g. a byte if $q = 2^8$ or a bit if $q = 2$).

5.1.2 Analytical Characterization of MS RLNC

We discuss the decoding delay of the macro-symbol based shifting schemes through the decoding probability analysis, as well as the computational complexities of the encoding and decoding operations.

5.1.2.1 Decoding Probability

We note that this part appeared in [4], and has not been edited for the sake of the clarity of the analysis. "MS RLNC multiplies the coding coefficients $\alpha_{\eta i},\ i = 1, 2, \ldots, N$, with the original macro-symbols in the column position λ (or the original macro-symbols that have been shifted into the column position λ) in (packet) rows i to compute coded macro-symbol $c_{\eta\lambda}$. We define $\mathcal{A}_\lambda$ as the set of coding coefficients that are actually involved in the computation of coded macro-symbol $c_{\eta\lambda}$, i.e., the coefficients that have encountered an actual original macro-symbol, and we define these coding coefficients as *actual coding coefficients*. We denote $a_\lambda = |\mathcal{A}_\lambda|$ for the number of actual coding coefficients involved in computing coded symbol $c_{\eta\lambda}$. Furthermore, we define $a_{\ell\lambda} = |\mathcal{A}_\lambda \cap (\bigcup_{1 \le \ell < \lambda} \mathcal{A}_\ell)|$ as the number of actual coding coefficients that were involved in computing coded symbol $c_{\eta\lambda}$ and in computing any of the preceding coded symbols $c_{\eta\ell},\ 1 \le \ell < \lambda$, in the considered coded packet η. For instance, in the example in Figure 5.3, $a_1 = 2$, $a_2 = 2$, and $a_{\ell 2} = 1$.

We define $E_\lambda^{(K)},\ \lambda = 1, 2, \ldots, \Lambda_{\max}$, as the event of decoding the $\Delta_\lambda^{\text{det}}$ coded MSs in column position λ after the receipt of the K-th coded packet (see Figure 5.4). The event $E_1^{(K)}$ of decoding the coded MSs in column position $\lambda = 1$, and the event $E_2^{(K)}$ for column position $\lambda = 2$, and so on, up to and including the event $E_{\Lambda_{\max}}^{(K)}$ for column position $\lambda = \Lambda_{\max}$ all joined together correspond to the event of decoding the entire generation, i.e., the complete set of N (shifted) source symbols after the receipt of coded packet K Thus, the probability of decoding the generation with K received coded packets is

$$P_{\text{dec}}^{(K)} = \mathbb{P}\left(\bigcap_{1 \le \lambda \le \Lambda_{\max}} E_\lambda^{(K)} \right). \tag{5.2}$$

For conventional packet based RLNC with GF(q), N source packets are decoded after

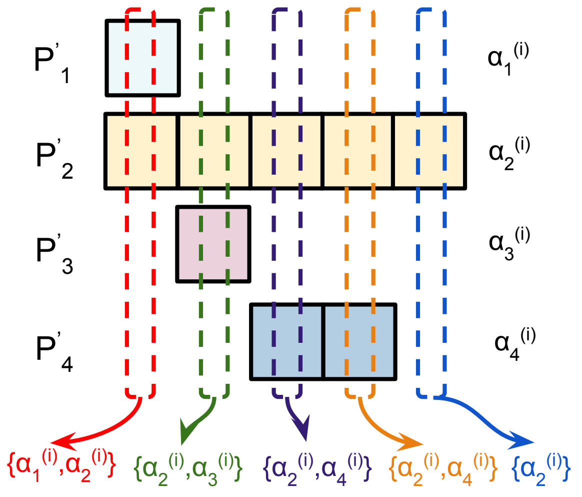


Figure 5.4: Example illustrating the determination of the set of coding coefficients that $\mathcal{A}_\lambda$ are actually involved in the computation of each coded macro-symbol at the column position $\lambda \in [\![1\ ;\ \Lambda_{\max}]\!]$.

the receipt of $K,\ K \geq N$, coded packets with probability [82, 83]:

$$\prod_{i=0}^{N-1} \left(1 - \frac{1}{q^{(K-n)}}\right). \tag{5.3}$$

In MS RLNC, there are Δ_1 macro-symbols in column position $\lambda = 1$, which were encoded with $a_1 = \Delta_1$ actual coding coefficients. Replacing N in Eq. (5.3) with a_1 gives the probability for decoding these $a_1 = \Delta_1$ macro-symbols from $K,\ K \geq \Delta_{\max}$, received coded packets, i.e.,

$$\mathbb{P}\left(E_1^{(K)}\right) = \prod_{n=0}^{a_1-1} \left(1 - \frac{1}{q^{(K-n)}}\right). \tag{5.4}$$

We proceed to evaluate the probability of decoding column 1 (event $E_1^{(K)}$) and column 2 (event $E_2^{(K)}$) with the conditional probability of decoding column 2 given that column 1 has already been decoded, i.e., we evaluate $\mathbb{P}\left(E_1^{(K)} \cap E_2^{(K)}\right) = \mathbb{P}\left(E_1^{(K)}\right) \cdot \mathbb{P}\left(E_2^{(K)} | E_1^{(K)}\right)$. The Δ_2 MSs in column position 2 have been encoded with a_2 actual coding coefficients. However, $a_{\ell 2}$ of these actual coding coefficients have been involved in the coding of the MSs in column 1, which have already been decoded. Thus, there are only $a_2 - a_{\ell 2}$ new coding coefficients that have not been involved in decoding the preceding columns. Thus, only $a_2 - a_{\ell 2}$ new coding coefficients need to effectively be considered in the decoding of the present column. However, there are also effectively only $K - (a_2 - a_{\ell 2})$ new received coded packets available that have not been involved in the decoding of the preceding columns. Hence, the conditional decoding probability is

$$\mathbb{P}\left(E_2^{(K)}|E_1^{(K)}\right)=\prod_{n=0}^{a_2-a_{\ell 2}-1}\left(1-\frac{1}{q^{(K-(a_2-a_{\ell 2})-n)}}\right). \tag{5.5}$$

Continuing this reasoning for the subsequent columns up to and including column $\Lambda_{\max}$ gives

$$P_{\text{dec}}^{(K)}=\mathbb{P}\left(E_1^{(K)}\right)\times\cdots\times\mathbb{P}\left(E_{\Lambda_{\max}}^{(K)}\middle|\bigcap_{\lambda<\Lambda_{\max}}E_\lambda^{(K)}\right). \tag{5.6}$$

Thus, with product notation,

$$P_{\text{dec}}^{(K)}=\prod_{\lambda=1}^{\Lambda_{\max}}\mathbb{P}\left(E_\lambda^{(K)}\middle|\bigcap_{\nu<\lambda}E_\nu^{(K)}\right), \tag{5.7}$$

whereby, analogous to Eq. (5.5),

$$\mathbb{P}\left(E_\lambda^{(K)}\middle|\bigcap_{\nu<\lambda}E_\nu^{(K)}\right)=\prod_{n=0}^{a_\lambda-a_{\ell\lambda}-1}\left(1-\frac{1}{q^{(K-(a_\lambda-a_{\ell\lambda})-n)}}\right). \tag{5.8}$$

Therefore, combining Eqs. (5.7) and (5.8),

$$P_{\text{dec}}^{(K)}=\prod_{\lambda=1}^{\Lambda_{\max}}\prod_{n=0}^{a_\lambda-a_{\ell\lambda}-1}\left(1-\frac{1}{q^{(K-(a_\lambda-a_{\ell\lambda})-n)}}\right). \tag{5.9}$$

” [4].

5.1.2.2 Numerical Verifications

We investigate the decoding probability related to the example given in Figure 4.1 as a function of the number of coded packets needed in excess of $\Delta_{\max}^{\text{det}}$, which represents the optimal performance, as well as the finite field choice. We run 10^4 simulations for every scheme (without considering packet losses) in order to exclusively focus on the code structure of this stage. The benefits of needing fewer packets to decode are compounded when losses are introduced into the system. Figure 5.5 displays a perfect matching between the analytical results and our encoding/decoding shifting schemes, with a minor difference of the order of 10^{-3}. For instance, around 10% of the generation is decoded after two transmissions $K = 2$ for the deterministic, thus we approve the advocated

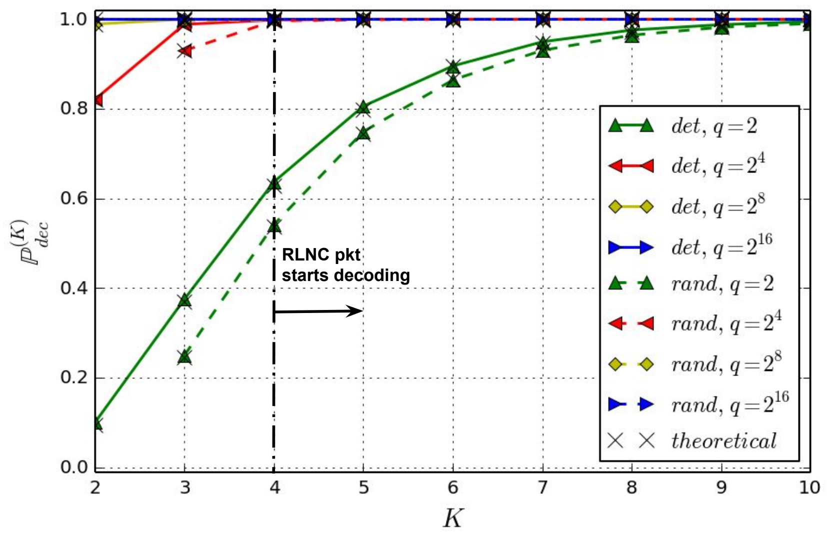


Figure 5.5: Probability of decoding for different finite fields in terms of number of coded packets to recover the generation of $N = 4$ packets in Figure 4.1, where "det" refers to the "deterministic shifting", "rand" refers to the "random shifting", and "theoretical" refers to the analytical bounds.

probability of decoding $P_{\text{dec}}^{(K)}$.

5.1.2.3 Computational Complexity

The encoding complexities for the random shifting and the deterministic schemes are similar to the encoding complexity of the previously discussed MS RLNC approaches in Chapter 4. Thus, the overall encoding complexity is of the order

$$\mathcal{O}(\mu \Lambda_{\max} \Delta_{\max}^{\text{rand/det}}). \tag{5.10}$$

As for decoding, for each subsequent column $\lambda,\ \lambda = 2, 3, \ldots, \Lambda_{\max}$, the decoding is simplified if the set $\mathcal{A}_\lambda$ of actual coding coefficients for column λ is identical to one of the sets of the actual coding coefficients for any of the preceding columns $\mathcal{A}_\ell,\ 1 \le \ell < \lambda$. If $\mathcal{A}_\lambda = \mathcal{A}_\ell$ for some prior column $\ell,\ 1 \le \ell < \lambda$, then only the final Gaussian elimination step with complexity contribution $\mathcal{O}(\Delta_\lambda^2) = \mathcal{O}(N^2)$ needs to be completed. Thus, in the best case scenario when $\mathcal{A}_\lambda = \mathcal{A}_1$ for all $\lambda,\ \lambda = 2, 3, \ldots, \Lambda_{\max}$, the computational complexity of MS RLNC decoding is

$$\mathcal{O}(N^3 + (\Lambda_{\max} - 1)N^2) = \mathcal{O}(N^3). \tag{5.11}$$

For instance, in the example in Figure 5.2, the complexity contribution is $\mathcal{O}(N^3)$ for column $\lambda = 1$ of the source symbols (on the left), or equivalently for the first row of the decoding matrix (on the right). Similarly, column $\lambda = 2$ of the source symbols (or equivalently the second row of the decoding matrix) incurs a decoding complexity contribution $\mathcal{O}(N^3)$. However, column $\lambda = 3$ of the source symbols (third row of the decoding matrix) has the same coding coefficient (the four) as the preceding source symbol column (row of the decoding matrix); thus incurring only a decoding complexity of $\mathcal{O}(N^2)$.

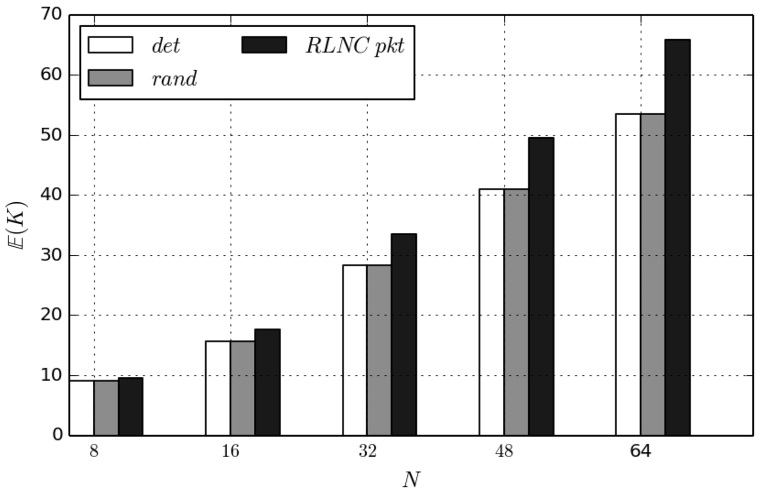
5.1.3 Performance Evaluation

We evaluate the random and deterministic shifting schemes through the number of coded packets K needed to decode a generation of size N source packets. Our results are based on a point-to-point topology as well as three-node topology to showcase the impact of recoding on our proposed schemes.

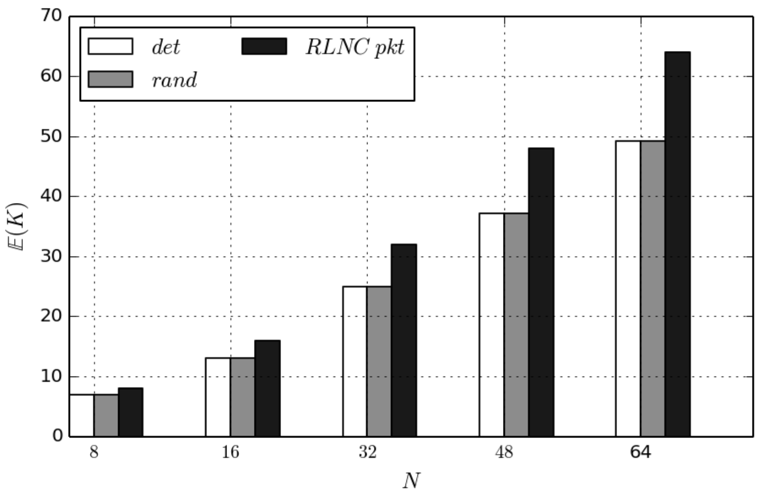
5.1.3.1 In a Point-to-point Network

Figure 5.6 shows that on average, both schemes behave similarly. Independent of the field size, the more a generation grows in size the less coded packets are required for decoding compared to *RLNC pkt* (the conventional RLNC). For example, around 23% of the overhead due to zero-padding is eliminated for $N = 64$, i.e., the overhead effect is basically eliminated.

Figure 5.7 depicts the CDF of the VP9 (HD) trace's portion using the shifting schemes when $q = 2$ and $q = 2^8$. At first glance, we notice that decoding is achieved for both schemes even before conventional RLNC starts with a probability close to one. We proved that 30% of padding overhead results from performing RLNC on such a trace [3, 4], is not only extremely reduced but also the varying sizes were exploited intelligently. This has allowed diminishing the number of received coded packets before decoding to $K = 25$ when using high field size, on the deterministic scheme with a probability close to 0.45, and the random shifting scheme with a probability close to 0.24. Additionally, the gap between the improvement brought by these schemes compared to RLNC is important even for the binary field. Such optimization is, however, finite field-dependent like any other RLNC strategy.



(a) $q = 2$



(b) $q = 2^8$

Figure 5.6: Comparison of the mean number of coded packets needed to decode a portion of a VP9 (HD) video trace for different generation sizes using different finite fields, $\mu = 60$ i.e. 25 macro-symbols per packet.

5.1.3.2 In Two-hop Network

To show the recoding capabilities of the system in a multihop setup and its ability to decode in the presence of random losses, we simulated a two-hop network topology of a sender, a relay, and a destination. Figure 5.8 shows the mean total number of packets transferred by the system in the aforementioned setup in the presence of random losses on both links when using the deterministic scheme and a traditional RLNC scheme. Each link has an error probability, ϵ_1 and ϵ_2 correspondingly. For Figure 5.8 we fixed ϵ_1 to 0.1,

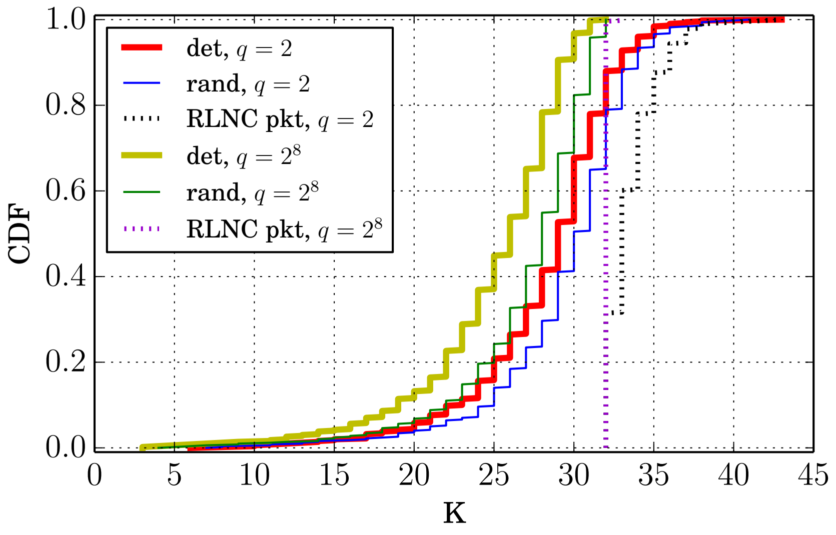


Figure 5.7: CDF comparison of the number of coded packets needed to decode 16×10^3 packets from an FHD video trace for the finite fields sizes $q = 2$ and $q = 2^8$, $N = 32$ and $\mu = 60$, i.e., 25 MS per packet.

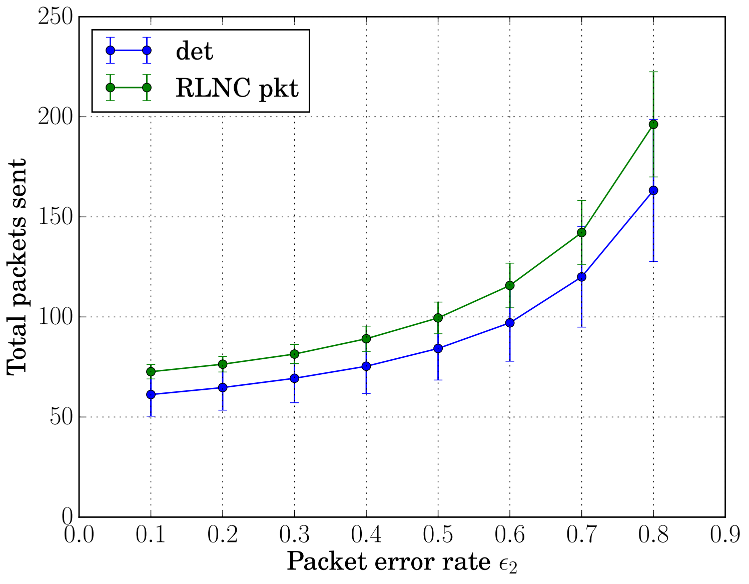


Figure 5.8: Mean total packets sent by the system in a two-hop topology in the presence of errors with probabilities $\epsilon_1 = 0.1$ and ϵ_2 correspondingly.

and varied the value of ϵ_2 in the range $\{0.1, .., 0.8\}$, and set $N = 32$ for both schemes. We notice that the mean total number of packets transferred over the network is lower in our scheme than in the RLNC scheme. The reason for this is that an RLNC strategy always needs to convey at least N packets to the destination. On the other hand, our proposed scheme requires fewer packet transmissions per link.

5.2 Evaluation of Padding Overhead RLNC Approaches

We evaluate the performance of all state-of-the-art schemes, as well as our proposed MS RLNC approaches against a naive adoption of the zero-padding as a solution to eliminating the unequal-sized packets encoding and decoding issues in RLNC. We note that based on the recommendations in [169], we set the largest packet size of the simple bundling, and the fragmentation size of chaining and fragmentation to the largest packet size $L_{\max}$ of a given generation. To do so, we consider in the following the transmission of excerpts of 16,000 video frames from the benchmark VP9 (HD) and the SVC (CIF) video traces, see Chapter 3. Moreover, we set the macro-symbol size to $\mu = 60$ bytes based on the results in Chapter 4. This means that one macro-symbol corresponds to 480 bits or 60 symbols in GF(2^8).

The metrics considered in the following comparisons are the *computational complexity*, *padding overhead*, and *number of coded packets*. For instance, we evaluate the overhead percentage, which is the total amount of padding overhead (bytes) to the total amount of data (bytes) that needs to be transmitted over an end-to-end network for each generation consisting of N variable size packets. We report the overhead for all generations using box plots for a complete and fairer comparison. Moreover, we evaluate the number of coded packets K required to complete the RLNC decoding at the receiver of each of the generations constituting the considered video trace portion. We report the K values for the individual generations in box plots. The K values represent a delay measure, i.e., the total number of coded packet transmissions required to transport a generation of video packets to the receiver. Finally, we remind that the results of this section appeared first in [4].

5.2.1 Computational Complexity

The complexities of the MS RLNC schemes are summarized in Table 5.1 and are compared to the existing approaches.

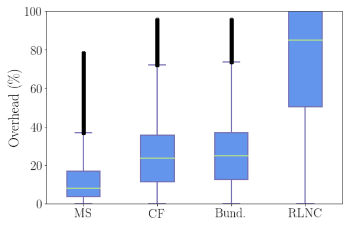
Table 5.1: Comparison of encoding and decoding complexity for a generation consisting of N packets. The number of unknowns indicates the number of unknown elements during RLNC decoding. The minimum number of coefficients indicates the minimum number of coding coefficients required for RLNC decoding (in absence of linear dependencies of coding coefficients and losses) [4].

Scheme	# Unknowns	Min. # coef	Encoding	Decoding
Conv. RLNC	N	N^2	$\mathcal{O}(L_{\max}N^2)$	$\mathcal{O}(N^3)$
Simple bundl.	N_{SB}	N_{SB}^2	$\mathcal{O}(N^2 + L_{\max}N_{\text{SB}}^2)$	$\mathcal{O}(N_{\text{SB}}^3)$
Chain & Fragm.	N_{CF}	N_{CF}^2	$\mathcal{O}(L_{\max}N_{\text{CF}}^2)$	$\mathcal{O}(N_{\text{CF}}^3)$
Progr. short.	$\sum_{\lambda=1}^{\Lambda_{\max}} \Lambda_\lambda$	N^2	$\mathcal{O}(\mu\Lambda_{\max}\Delta_{\max})$	$\mathcal{O}(\Delta_{\max}^3) - \mathcal{O}(\Lambda_{\max}\Delta_{\max}^3)$
Random shift.	$\sum_{\lambda=1}^{\Lambda_{\max}} \Lambda_\lambda$	$\Delta_{\max}^{\text{rand}^2}$	$\mathcal{O}(\mu\Lambda_{\max}\Delta_{\max}^{\text{rand}})$	$\mathcal{O}(\Delta_{\max}^{\text{rand}^3}) - \mathcal{O}(\Lambda_{\max}\Delta_{\max}^{\text{rand}^3})$
Det. shift.	$\sum_{\lambda=1}^{\Lambda_{\max}} \Lambda_\lambda$	$\Delta_{\max}^{\text{det}\ 2}$	$\mathcal{O}(\mu\Lambda_{\max}\Delta_{\max}^{\text{det}})$	$\mathcal{O}(\Delta_{\max}^{\text{det}\ 3}) - \mathcal{O}(\Lambda_{\max}\Delta_{\max}^{\text{det}\ 3})$

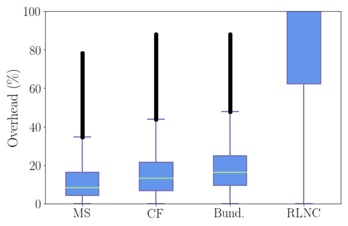
5.2.2 Padding Overhead

Figure 5.9 illustrates in the form of box plots the padding overhead for the considered generations, containing each N consecutive video packets. The box plots marked with "MS" represent all the macro-symbol-based approaches that we proposed in this dissertation because the macro-symbols create a fixed padding overhead regardless of the scheme adopted. We note that the random shifting approach achieves similar results if it is combined with a shortening policy. We remark that the macro-symbol approaches generate significantly less padding overhead than the other considered schemes, namely, chaining and fragmentation (CF) and simple bundling (Bund) for the small generation sizes $N = 4$ and $N = 8$. This difference shrinks, however, as the generation sizes increase. All the macro-symbol approaches as well as the ones proposed in [169] have similar padding overhead, as is the case for $N = 16$. We note that we did not add the additional evaluations for larger generation sizes to avoid clutter.

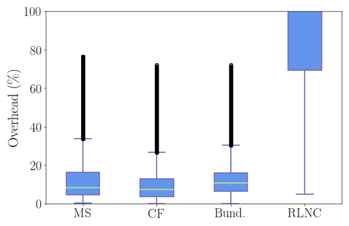
The bundling, as well as the chaining and fragmentation schemes encode equal packets of size $L_{\max}$ each. The zero-padding arises in chaining and fragmentation due to the long chain of packets that have a cumulative length of $\sum_{n=1}^{N} L_n$, which is not necessarily an integer multiple of $L_{\max}$. There are $\lfloor \sum_{n=1}^{N} L_n / L_{\max} \rfloor$ portions of size $L_{\max}$, and a last one of size $\sum_{n=1}^{N} L_n \mod L_{\max}$, which needs to be padded with $L_{\max} - (\sum_{n=1}^{N} L_n \mod L_{\max})$ zero symbols before the encoding operations. The padding overhead is on average $L_{\max}/2$ if we assume a uniform distribution of the size of the last portion size.



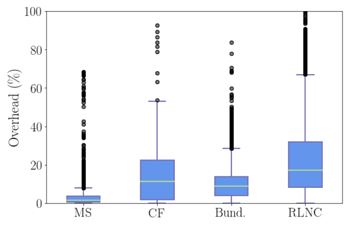
(a) SVC (CIF), $N = 4$



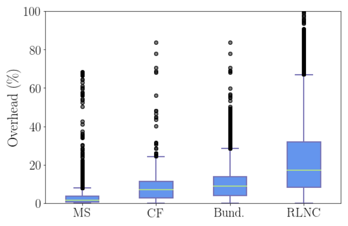
(b) SVC (CIF), $N = 8$



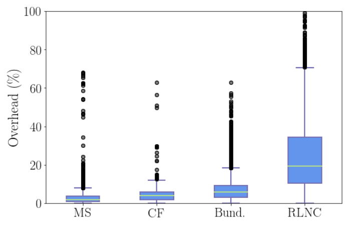
(c) SVC (CIF), $N = 16$



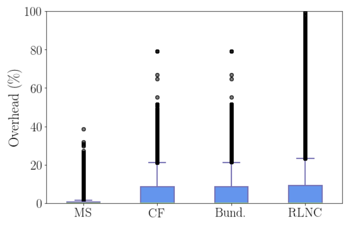
(d) VP9 (HD), $N = 4$



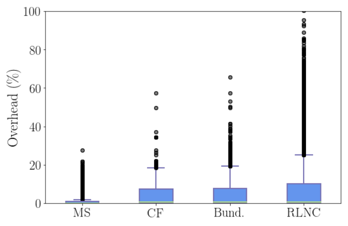
(e) VP9 (HD), $N = 8$



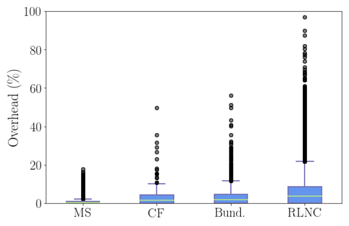
(f) VP9 (HD), $N = 16$



(g) H.265 (4k), $N = 4$



(h) H.265 (4k), $N = 8$



(i) H.265 (4k), $N = 16$

Figure 5.9: Box plot comparison of the padding overhead percentage for different generation sizes N. Fixed parameters: GF(2^8), $\mu = 60$ bytes, i.e., 25 MSs per packet [4].

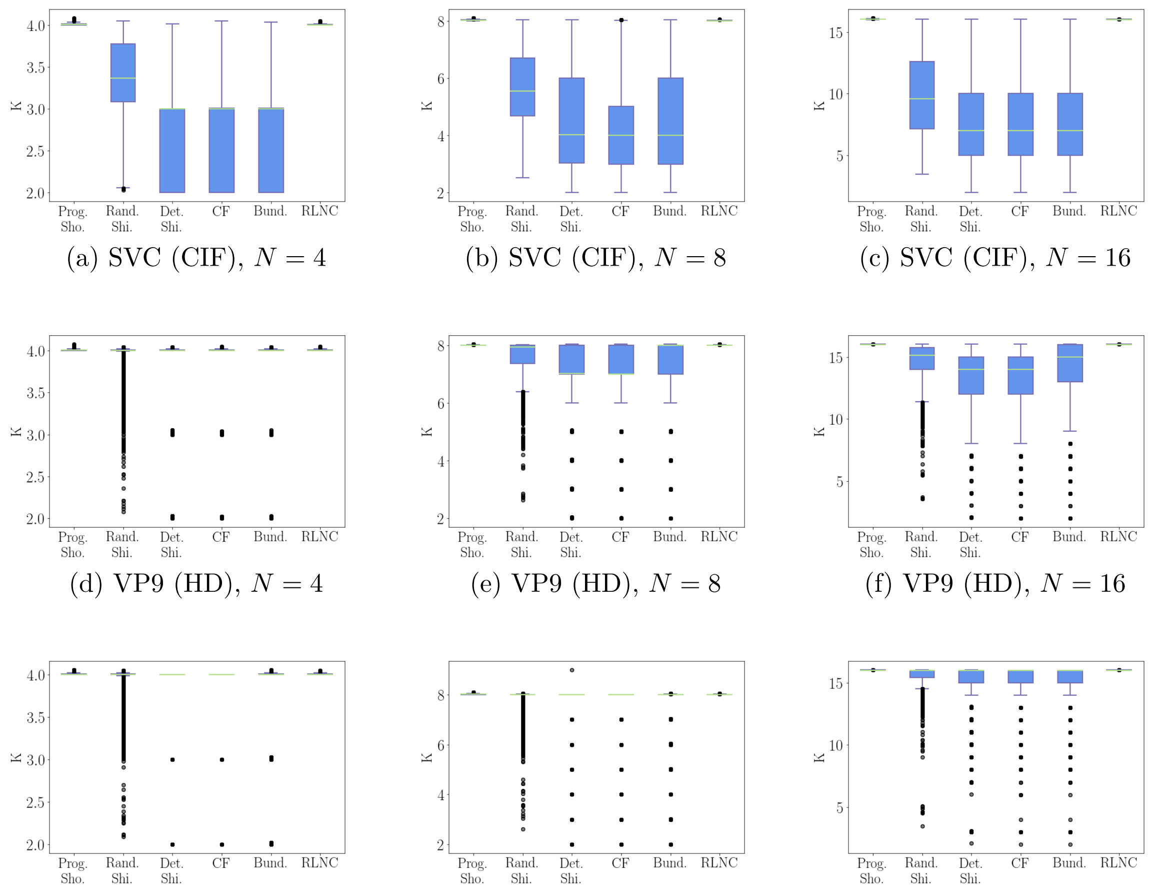
Accordingly, the equivalent average padding overhead ratio is $L_{\max}/(2\sum_{n=1}^{N} L_n)$, which is inversely proportional to the generation size N. We also note that decreasing the fragmentation size F_s considerably compared with $L_{\max}$ would lead to a spike in the number of resulting packets. This is not a viable strategy as it simply increases the number of coding coefficients required, thus the computational complexities, and the energy consumption to communicate the coded packets. On the other hand, the padding overhead percentage of the simple bundling approach is, similarly, inversely proportional to the size of the generation N. "On the other hand, the macro-symbol approaches pad every packet to an integer multiple of the macro-symbol size μ. If we assume that the packet sizes are integer multiples of μ plus a "spill-over" part that is uniformly distributed over μ, the average padding overhead is $\mu/2$ per packet. The corresponding percentage overhead is $N\mu/(2\sum_{n=1}^{N} L_n)$, which is independent of the generation size N" [4].

We remind that the deterministic shifting approach firstly pads the packets to an integer number of macro-symbols, then shifts the resulting packets at the granularity of macro-symbols, as explained earlier in this chapter. An alternative approach could start by shifting the packets at the granularity of individual symbols (bits in GF(2)), analogous to the chain and fragmentation approach. The shifted packets could be afterward fragmented into macro-symbols (whereby a given MS may contain parts of multiple packets) and encoded using macro-symbol RLNC. This alternative approach guarantees that the only "spillover" into the last macro-symbol padded out to a full macro-symbol occurs in the last row. This decreases the average padding overhead to $\mu/2$ for the entire generation. With such an approach, one has to signal the packet sizes at the granularity of symbols, thus needing up to $N \log_2 L_{\max}$ bits for the generation, whereas the deterministic shifting approach is only required to signal the packet lengths at the granularity of macro-symbols, i.e., up to $N \log_2 \Lambda_{\max}$ bits (the difference of $N \log_2 \mu$ bits is considered negligible in most cases).

From Figure 5.9 we observe that the SVC (CIF) video requires higher padding overhead generally compared to the VP9 (HD) video, for all the various padding reduction techniques without an exception. This is because the SVC (CIF) video trace has a higher probability of packet sizes below the MTU, unlike the VP9 (HD) video (as shown in Chapter 3). We concluded in Chapter 3 that the padding overhead increases in general, with the tendency for the video to have smaller packets. Additionally, we remark from the previous results, as well as from Figure 5.9, that the padding overhead tends to increase with a larger generation size N. We conclude, therefore, from this comparison that the padding overhead reduction is considerably more critical when the generation size is large. Nevertheless, the macro-symbol-based approaches are more effective than the state-of-the-art approaches for small generation sizes N. Moreover, for moderate to large generations, all of the schemes we discussed perform approximately similarly.

5.2.3 Number of Coded Packets

We evaluate using two-channel conditions, with and without losses to understand if the losses could impact a specific scheme's overall number of coded packets needed to transmit. We maintain the same aforementioned settings.



(a) SVC (CIF), $N = 4$ (b) SVC (CIF), $N = 8$ (c) SVC (CIF), $N = 16$

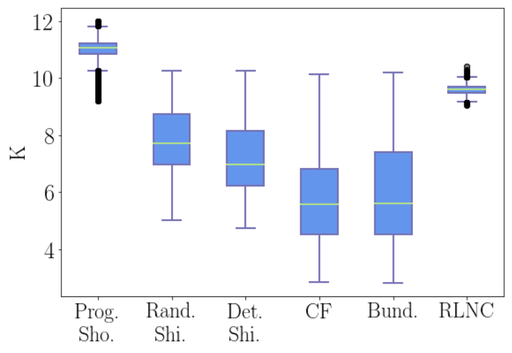
(d) VP9 (HD), $N = 4$ (e) VP9 (HD), $N = 8$ (f) VP9 (HD), $N = 16$

(g) H.265 (4k), $N = 4$ (h) H.265 (4k), $N = 8$ (i) H.265 (4k), $N = 16$

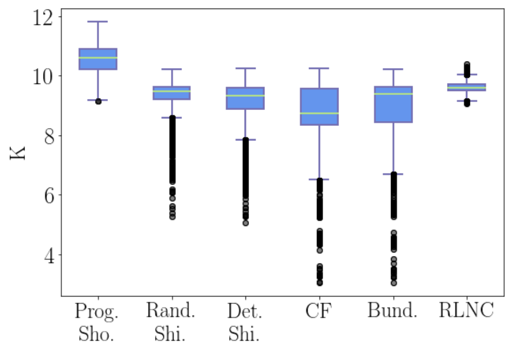
Figure 5.10: Box plot comparison of the number K of coded packets needed to decode the N packets in a generation. Fixed parameters: GF(2^8), $\mu = 60$ bytes, i.e., 25 MSs per packet, loss-free network link.

5.2.3.1 Without Packet Losses

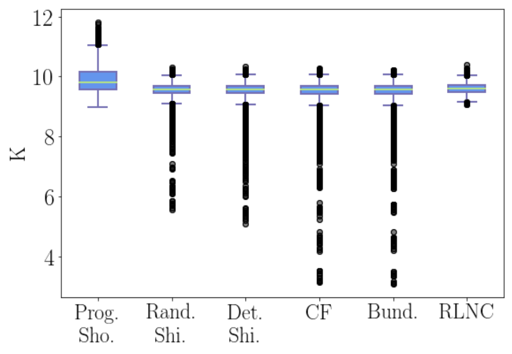
Figure 5.10 and Figure 5.11 compare the box plots for the number of coded packets K required to completely recover all the generations of size N formed from the portions of video frames considered. Figure 5.10 illustrates the box plots for GF(2^8) for generation sizes $N \in \{4, 8, 16\}$, whereas Figure 5.11 illustrates the box plots for GF(2) for $N = 8$. We observe from both figures that standard RLNC and the progressive shortening scheme require, on average, around $K = N$ coded packets for GF(2^8), but slightly more coded packets for GF(2). This is expected since the linear dependency among the coded packets increases with smaller finite fields, which explains the extra coded packets beyond N that are conveyed. However, we remind that although the standard RLNC and the progressive



(a) SVC (CIF)



(b) VP9 (HD)



(c) H.265 (4k)

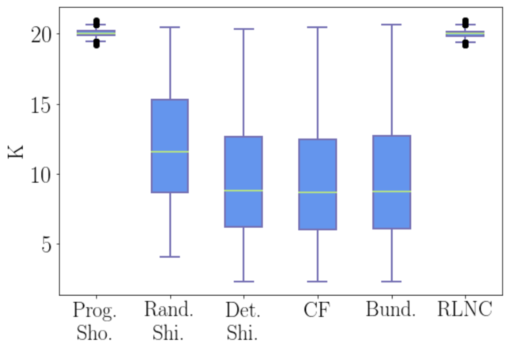
Figure 5.11: Box plot comparison of the number K of coded packets for GF(2) needed to decode the $N = 8$ packets in a generation. Fixed parameters: $N = 8$ generation size, GF(2), $\mu = 60$ bytes, i.e., 480 MSs per packet, loss-free network link.

shortening scheme transmit a similar number of coded packets, shorter packets require fewer transmission resources compared to full-size packets.

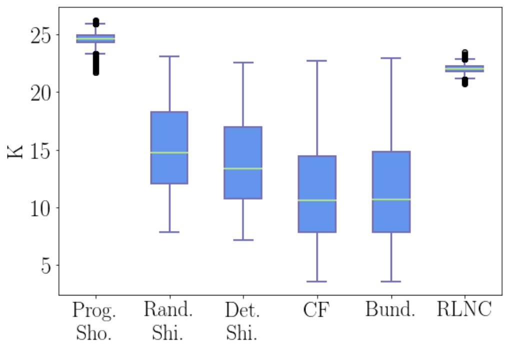
Furthermore, we remark from Figure 5.10 and Figure 5.11 that the deterministic shifting, as well as chaining and fragmentation, and bundling approaches tend to transmit the least number of coded packets K. The random shifting scheme tends, in general, to transmit slightly more coded packets, but this is still lower than the number of coded packets of conventional RLNC. Specifically, we observe from Figure 5.10 that the third quartiles of K for the deterministic shifting, as well as for chaining and fragmentation, and bundling are lower than the number of original packets N themselves, mainly for VP9 (HD) for $N = 16$ and all SVC (CIF) scenarios. This shows that these novel approaches can decode more than 75% of the generations before receiving N coded packets, whereas N coded packets is the bare minimum required for standard RLNC to successfully decode a generation. We also observe that the upper whiskers of K for the deterministic shifting, for chaining and fragmentation, and for bundling approaches for VP9 (HD) with $N = 16$, and for SVC (CIF) with $N = 8$ and 16 for GF(2^8) are at or below the minimum K values for conventional RLNC. This means that these approaches enable the decoding of 90% of the generations before conventional RLNC finishes decoding any generation. These lower numbers of coded packets K required with the padding reduction approaches indicate lower latencies for the network transport of the media packet generations.

These reductions in terms of the number of coded packets required are achieved through the “packing” of the N variable size packets into fewer full-sized packets. Particularly, these padding reduction schemes judiciously exploit the heterogeneous packet sizes to “pack” the data belonging to one generation into fewer packets, which are later used for RLNC encoding and transmission of newly designed generations. On the other hand, the random shifting scheme packs the data somewhat less effectively compared to the other approaches. This is mainly due to the fact that random shifting does not always guarantee minimizing the length of the longest column to encode.

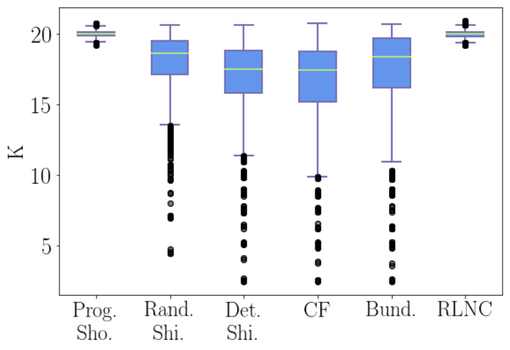
“Large generation sizes N and videos with a significant number of small packets provide extensive opportunities for the “packing” of unequal size packets into fixed-size packets. Therefore, we observe more pronounced reductions of the number K of required coded packets for the SVC (CIF) video (with a substantial portion of small frames) than for the VP9 (HD) video. Also, the large considered generation size of $N = 16$ exhibits more pronounced reductions of K than the smaller $N = 4$ and 8 generation sizes. We verified in additional evaluations that are not included to avoid clutter that generation sizes above 16 further slightly increase the packing opportunities. For instance, for VP9 (HD) for $N = 64$ and GF(2^8) the upper (90%) whisker of K for deterministic shifting MS RLNC is at 62” [4].



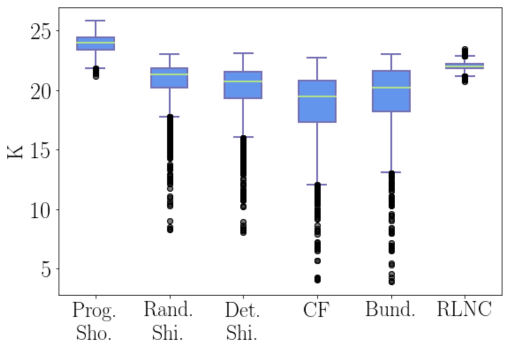
(a) SVC (CIF), GF(2^8)



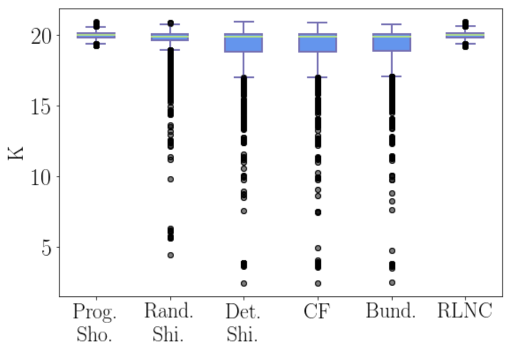
(b) SVC (CIF), GF(2)



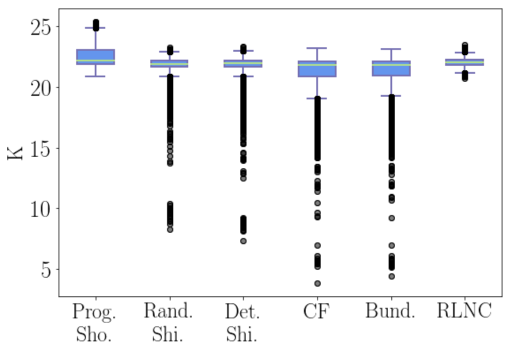
(c) VP9 (HD), GF(2^8)



(d) VP9 (HD), GF(2)



(e) H.265 (4k), GF(2^8)



(f) H.265 (4k), GF(2)

Figure 5.12: Box plot comparison of the number of required coded packet transmissions over a link with 20% packet loss to decode the $N = 16$ packets in a generation. Fixed parameters: $N = 16$ generation size, $\mu = 60$ bytes, i.e., 25 MSs per packet for GF(2^8) and 480 MSs per packet for GF(2).

5.2.3.2 With Packet Losses

We present the results that are related to the number of coded packet transmissions over a lossy link required to completely recover all the generations of $N = 16$ packets of the benchmark video traces. The considered link drops 20% of the conveyed coded packets independently at random. From Figure 5.12 we observe a similar performance behavior as in Figure 5.10 and Figure 5.11 despite considering a loss-free network link. The padding overhead reduction approaches continue to outperform the standard RLNC technique by requiring fewer coded packet transmissions. Such results are expected since these schemes are designed for preserving the RLNC features and mechanisms, including error correction. Simple bundling, as well as chaining and fragmentation, are basically pre-processing steps, which occur before the common RLNC encoding step at the sender's side, and are simply undone once decoding at the receiver is done. The progressive shortening approach makes, by design, the last macro-symbols of longer packets more susceptible to losses, especially when small finite fields are employed. This is because these last macro-symbols are less involved in creating coded packets compared to other macro-symbol-based RLNC schemes, which mainly generate full-length coded packets. The evaluation results displayed in Figure 5.12 reveal that this vulnerability effect of the progressive shortening is somewhat mild. We hardly notice a very slight increase in the median of the numbers of transmitted packets compared to the conventional RLNC with GF(2). As for GF(2^8), we observe a similarity in the medians of the progressive shortening and the standard RLNC. It is also worth reminding from Figure 5.9 that progressive shortening remains a favorable solution as it significantly reduces the padding overhead compared to conventional RLNC. The deterministic and random shifting approaches mainly operate on full-length packets, except for the last coded packet that has a size complying with the min-sized last coded packet policy described in Chapter 4. Accordingly, these shifting approaches have nearly similar packet decoding capabilities as the standard RLNC for any finite field considered. The evaluation results in Figure 5.12 particularly confirm that the deterministic shifting approach requires essentially the same low number of coded packets as the chaining and fragmentation over lossy links, while it incurs either similar or slightly lower padding overhead as previously shown in Figure 5.9.

When we closely compare Figure 5.12(a), (c), and (e) for the transmission over the lossy network link with the corresponding Figure 5.10(c), (f), and (i) to the transmission over the perfect network link, we notice that the losses slightly amplify the differences between the standard RLNC and the padding reduction schemes. Specifically, for the VP9 (HD) video, the deterministic shifting, and the chaining and fragmentation approaches have median K values of 14 compared to a median K value of 16 for standard RLNC in Figure 5.10(f), but median K values of 17.49 and 17.45, respectively, compared to 20 for the standard RLNC in Figure 5.12(c). Thes padding overhead reduction approaches are known to “pack” the unequal-sized packets of a certain generation into fewer packets to be communicated over a lossy network link. Transmitting fewer packets implies losing fewer packets, thus fewer additional transmissions are required to be able to solve an entire generation at the receiver. We conclude that the results in Figure 5.12 prove that the padding reduction approaches are effectively capable of reducing the number of coded packets to transmit over lossy network links compared to the standard RLNC, mainly for videos that have small frames.

5.2.3.3 Discussion of the Schemes Comparison

“We have conducted an extensive performance evaluation of padding reduction approaches with long traces of full-motion VBR video. We found that for small RLNC generation sizes that are required for low-latency video transmission, the MS RLNC approach with deterministic shifting reduces the padding overhead the most. For moderate to large RLNC generation sizes that incur moderate to large network transport delays, the chaining and fragmentation approach and the bundling approach of forming fixed-size packets from variable size packets and MS RLNC with deterministic shifting effectively reduce the padding overhead” [4]. We note that the extensively shifting of the original packets before generating a new coded packet resulted in a slow down of the simulations due to the very lengthy delays of decoding. That is why we decided not to include it in the dissertation. However, we would like to look into it in the future to understand the issue. This approach seems to increase the robustness of the code since we vary the shifts and the coding coefficients with every coded packet. Finally, since the performance of these shifting schemes is dependent on the used finite field, designing other schemes that overcome this issue will be the core of the following section.

5.3 Random Shift and XOR (RaSOR)

We have demonstrated the bottleneck in conventional and macro-symbol RLNC techniques, resulting from the cubic decoding computational complexity. This is prohibitive for power-constrained devices, like sensors. As we are shifting the distributed computing in IoT and mobile devices, the computational complexity along with the performance have both become critical factors that play a crucial role in the choice of coding strategies. Our idea is, therefore, to diminish the complexity of macro-symbol schemes, by simply not using any coding coefficients. We propose RaSOR, a macro-symbol and shifting approach that does not classify as RLNC based. This is the reason why we could not classify it with the previous approaches. It is, namely, based on an extensive pre-coding step of macro-symbol shifting, which is identical to the pre-coding of the random shifting scheme, and then only XOR operations are directly performed on the columns λ of the newly distributed macro-symbols. This approach could be easily and efficiently implemented on hardware and could be considered as an efficient code for multicast and broadcast transmissions, where recoding nodes are not needed. We note that the remaining of this section has first appeared in [30].

5.3.1 RaSOR: Random Shift and XOR Scheme

Without loss of generality, we propose the unicast transmission of a generation consisting of N original packets $P_1, P_2, \cdots, P_N$ of sizes $\Lambda_1, \Lambda_2, \cdots, \Lambda_N$ macro-symbols respectively [4]. The pre-coding step is exactly the same as the previously described pre-coding for the random shifting (see Figure 5.1).

5.3.1.1 Encoding

This scheme has two modes for encoding, namely, *random shift-and-XOR* and *systematic encoding*.

Random Shift-and-XOR Mode The encoding process is simply composed of two phases, namely, a random shifting of the packets [4], then XORing the macro-symbols available on the similar column λ. Normally, XORing does not work on its own as it either needs additional coding coefficients or different unknowns to create linearly independent elements. Scattering the macro-symbols at random before XORing creates possibilities for an innovative coded macro-symbols generation. In a nutshell, the encoding process consists of XORing vertically all macro-symbols that are shifted to the same position. Intuitively, there are exactly $\Lambda_{\max}$ coded macro-symbols to be stored in joint blocks to form a new coded packet, as illustrated in Figure 5.13. The largest packet P_2 has $\Lambda_{\max} = 5$ macro-symbols. Shifting and then XORing the newly arranged macro-symbols results in the coded packet C_η.

The coded macro-symbol $\mathbf{c}_{\eta\lambda}$, referring to the λ, macro-symbol of the coded packet C_η is the result of XORing the shifted input macro-symbols. Formally,

$$\mathbf{c}_{\eta\lambda} = \bigoplus_{\substack{1 \leqslant i \leqslant N \\ 1 \leqslant l \leqslant \Lambda_i}} s_{il}\zeta_{il}, \tag{5.12}$$

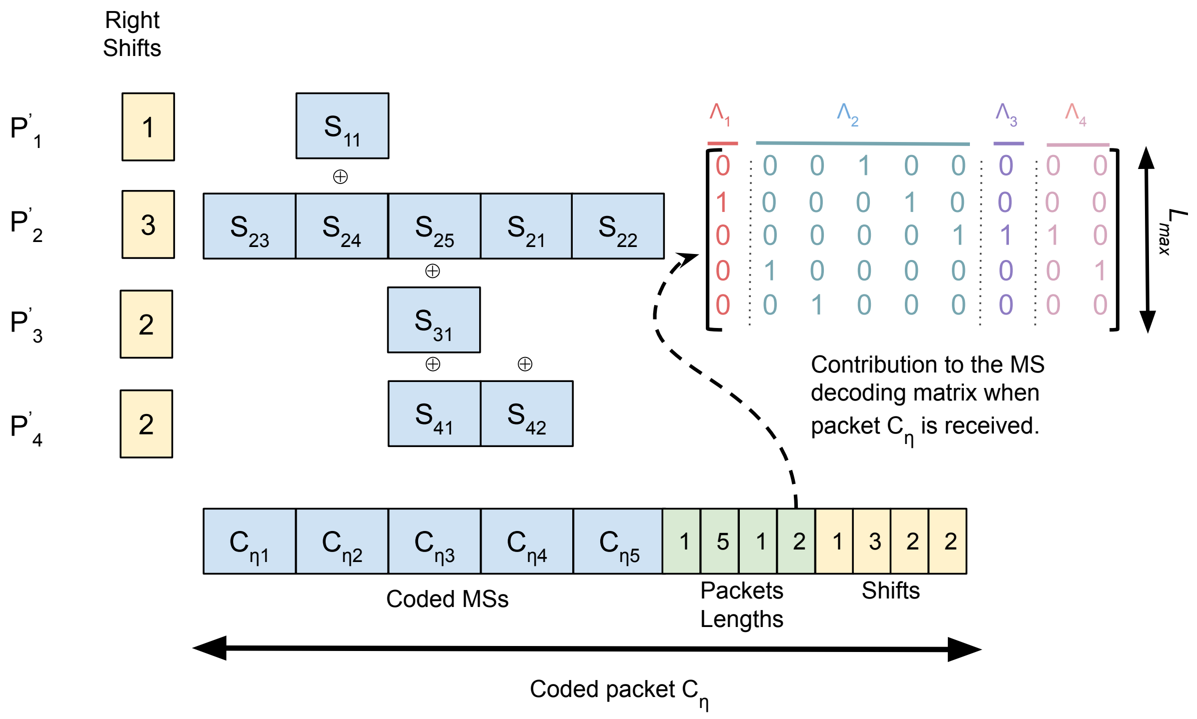


Figure 5.13: An example of the macro-symbol level encoding process after randomly shifting the packets to produce C_1, generation size $N = 3$, $\Lambda_{\max} = 4$.

where ζ_{il} is an indicator function that tells whether the input macro-symbol s_{il} is involved in generating the coded macro-symbol $\mathbf{c}_{\eta\lambda}$. ζ_{il} is defined as:

$$\zeta_{il} = \begin{cases} 1 & \text{if } l \in \mathcal{W}^{(i)} \\ 0 & \text{else,} \end{cases} \tag{5.13}$$

and $\mathcal{W}^{(i)} = W_1^{(i)} \cup W_2^{(i)}$ represents the set of all possible positions that a packet P_i could take after applying the random offset $\phi_l^{(i)}$, $i = 1, 2, \cdots, K$, and

$$\begin{aligned} W_1^{(l)} &= \{\phi_l^{(i)}, \cdots, (\phi_l^{(i)} + \Lambda_n)\} \quad \text{when } (\phi_l^{(i)} + \Lambda_n) \mod \Lambda_{\max} = 0 \\ W_2^{(l)} &= \{\phi_l^{(i)}, \cdots, (\phi_l^{(i)} + \Lambda_n)\} \cup \{1, \cdots, (\phi_l^{(i)} + \Lambda_n - \Lambda_{\max})\}. \end{aligned} \tag{5.14}$$

On the other hand, we denote by $\Delta_{\eta\lambda}$ the degree of the coded macro-symbol $\mathbf{c}_{\eta\lambda}$, i.e., the number of original macro-symbols XORed together to obtain $\mathbf{c}_{\eta\lambda}$. It represents the number of macro-symbols from different original packets put in a same position j, thus encoded together, here using XOR wise operations, where $1 \leqslant \Delta_{\eta\lambda} \leqslant N$. Based on

Eq. 5.12, the number of packets with contribution in $\mathbf{c}_{\eta\lambda}$ is:

$$\Delta_{\eta\lambda} = \sum_{\substack{1\leqslant i\leqslant N \\ 1\leqslant l\leqslant \Lambda_i}} \zeta_{il}. \tag{5.15}$$

The design of this scheme depends on shifting, thus knowing the positions of the macro-symbols is crucial. As a result, we need to communicate the sizes as well as the shifts of the original packets. There is no need for an encoding vector, and the sizes could be transmitted only once if the channel is perfect. However, every coded packet must come along with the shifts, as these vary with every new coded packet created.

Systematic Encoding Mode *Systematic network coding* was first proposed in [86,185], where nodes forward or replicate the packets aside from sending coded versions, to avoid the reduction of the system's throughput due to the computations of a heavy linear combination among others. Furthermore, it was implemented using $GF(2)$ in battery-constrained mobile devices known to have low computational capabilities. Lucani et al. analyzed it from computational complexity and field size perspectives in [87]. It was also adopted in scenarios related to erasures, and the linear dependency brought by the use of small finite fields and the computational complexity. In order to comply with the design of the previously presented schemes in this dissertation, we propose to employ the systematic coding approach on macro-symbols. As such, a coded packet is simply a set of $L_{\max}$ consecutive macro-symbols transmitted using the systematic mode. We note that such a mode requires minimal coding overhead when used on its own. Thus we communicate the packet sizes within the first transmission. Nonetheless, we send the macro symbols index, δ_i, starting from systematic packet C_i, that is created for enhanced decoding reliability. The utility of this index will be further discussed in this chapter.

5.3.1.2 Decoding: On-the-Fly XORing

The benefits of using on-the-fly decoding are that it enables the destination to start the decoding of a generation from the very first received coded packet, because the first coded packet itself comprises $\Lambda_{\max}$ coded macro-symbols. The coding vector contains solely the original packet sizes in the number of macro-symbols and random shifts that vary with every coded packet. This information is expanded into the decoding matrix of size $\sum\limits_{i=1}^{N} \Lambda_i \times N$ (see the rightmost part of Figure 5.13). This matrix has exactly $\sum\limits_{i=1}^{N} \Lambda_i$ "1" bits coefficients, i.e., $1\big/\Lambda_{\max}$ of the matrix's size is just one bit elements. The positions of these ones vary with every transmission, following a certain pattern that obeys the random shifts and the packets' sizes. We propose to use on-the-fly Gaussian elimination [62], where we insert every row of the matrix related to coded packet C_i into the overall decoding matrix of size $\sum_{i=1}^{N} \Lambda_i \times \sum_{i=1}^{N} \Lambda_i$, if and only if it has a pivot that was not identified previously. We note that graph-based mechanisms, such as the belief propagation, could be adopted in such types of coding where only XORed elements are in play. Eventually, the use of macro-symbols may provide a way to do parallel computation.

5.3.2 Decoding Analysis and Regions of Operations

We discuss the conditions under which the random shift-and-XOR mode of the RaSOR scheme is capable of generating coded packets that could be solved by the decoder. We provide a mathematical framework that explains this limitation, and we propose some potential policies, namely the hybrid encoding and the macro-symbol policies for equal packets.

5.3.2.1 Condition

The shift-and-XOR mode of the RaSOR scheme, unfortunately, cannot be applied to generations that have more than exactly one *full-length packet*, i.e., a packet of size $\Lambda_{\max}$ MSs. This limitation is due to the constrained design which only allows the coding exclusively inter-packets. For example, two or more macro-symbols from the same source packet cannot be XORed together. To better understand this disadvantage, we display the source packets as independent sets, and their respective macro-symbols constitute the corresponding elements, as follows:

$$P_i = \{s_{ij}, j \in \mathcal{L}_j = \{1, \cdots, \Lambda_i\}, i = 1, 2, \cdots, K\}, \tag{5.16}$$

where $\Lambda_i = Card(P_i) = |P_i|$. We propose to count the possible number of linear combinations obtained when we choose at random at most one element from each set to XOR together (some packets might be shorter, so they cannot be involved in all coded macro-symbols). This is analogous to the random shift-and-XOR mode, i.e., we apply a random shifting vector on the packets before XORing the macro-symbols of the same positions together. We note that the following definitions and propositions previously appeared in [30].

Equal-sized packets This is the case where all the possible combinations should have exactly N elements. This means that exactly one element is picked at random from each set. Thus, the number $\mathcal{C}$ of all different XOR combinations obtained if we use up all shifting possibilities is according to [30]

$$\mathcal{C} = {\Lambda_{\max}}^{N}. \tag{5.17}$$

Unequal-sized packets This case refers to the encoding of packets that have one or more original packets that are not full-length. We denote the set of indices of full-length packets $\mathcal{K}$, i.e.,

$$\mathcal{K} = \{i : \Lambda_i = \Lambda_{\max}, i \in [\![1 \; ; \; N]\!]\},$$

$\mathcal{K}^c$ its complementary set, and $|\mathcal{K}|$ and $|\mathcal{K}^c|$ their respective cardinals. Specifically, a set with $|\mathcal{K}^c|$ could participate with one element in the combinations or could simply not be chosen to participate. Accordingly, we express $\mathcal{C}$ as:

$$\mathcal{C} = \Lambda_{\max}^{|\mathcal{K}|}\Big[1 + \sum_{i\in\mathcal{K}^c} \Lambda_i + \sum_{\substack{i_1, i_2 \in \mathcal{K}^c \\ i_2 \neq i_1}} \Lambda_{i_1}\Lambda_{i_2} + \cdots + \prod_{i\in\mathcal{K}^c} \Lambda_i\Big]. \tag{5.18}$$

Definition 2. *[30] We define the maximum rank that a linear system consisting of $i\Lambda_{\max}$ equations could have, i.e. the maximum rank evolution after receiving exactly i coded packets, each of which has exactly $\Lambda_{\max}$ macro-symbols:*

$$\begin{aligned} \max rk(i\Lambda_{\max}) &= \max rk\Big\{C_1, \cdots, C_i\Big\} \\ &= \max rk\Big\{c_{11}, \cdots, c_{1\Lambda_{\max}}, \cdots, c_{i1}, \cdots, c_{i\Lambda_{\max}}\Big\}. \end{aligned} \tag{5.19}$$

Proposition 1. *[30] The maximum number of independent linear equations in a linear system of equations constructed by XORing randomly chosen elements from N sets with the same cardinality is*

$$\max rk(\mathcal{C}) = N\Lambda_{\max} - (N-1). \tag{5.20}$$

Proof. The proof is available in Appendix A. □

Proposition 2. *[30] If at least two packets have size $\Lambda_{\max}$ in a generation, i.e., $|\mathcal{K}| \geqslant 2$, then the linear system of equations can never be full rank, regardless of all possible shift* *combinations used, and the maximum rank it could reach is*

$$\max rk(\mathcal{C}) = \sum_{i=1}^{N} \Lambda_i - (|\mathcal{K}| - 1). \tag{5.21}$$

Proof. The proof is available in Appendix A. □

Definition 3. *Let Δ_{opt} be the minimum number of coded packets needed in total to recover a generation, i.e. the minimum number of macro-symbols that constitute a maximum of full-size packets:*

$$\Delta_{\text{opt}} = \left\lceil \sum_{i=1}^{N} \Lambda_i \Big/ \Lambda_{\max} \right\rceil = \Delta_{\text{det}}. \tag{5.22}$$

In addition, we prove that decoding starting from the bare minimum Δ_{opt} is not possible for the cases where $\Delta_{\text{opt}} \mid \Lambda_{\max}$, i.e., Δ_{opt} is a divisor of $\Lambda_{\max}$.

Proposition 3. *The probability that the linear system of equations obtained after receiving i coded packets is full rank, i.e. equals to $iL_{\max}$ is*

$$\mathbb{P}\{rk\Big\{C_1, \cdots, C_i\Big\} = i\Lambda_{\max}\} = \begin{cases} 1 & \textit{if } i = 1 \\ 0 & \textit{else} \end{cases} \tag{5.23}$$

and,

$$\max rk\Big\{C_1, \cdots, C_i\Big\} = i\Lambda_{\max} - (i-1), \quad \forall i > 1 \tag{5.24}$$

Therefore, if $\Delta_{\text{opt}} \mid \Lambda_{\max}$ then we need at least $K = \Delta_{\text{opt}} + 1$ coded packets to decode, and the maximum rank achieved when obtaining $\Delta_{\text{opt}}\Lambda_{\max}$ linear equations is

$$\max rk(\Delta_{\text{opt}}\Lambda_{\max}) = \sum_{i=1}^{N} \Lambda_i - 1. \tag{5.25}$$

Proof Sketch. We prove it by contradiction. Suppose that 2 coded packets are innovative. Then, $2\Lambda_{\max}$ linear equations related to the coded macro-symbols are independent, i.e. $rk\{C_1, C_2\} = 2\Lambda_{\max}$. However, XORing any $2L_{\max} - 1$ equations is equal to the remaining one, which contradicts the hypothesis. □

5.3.2.2 Hybrid Encoding

We proved that the random shift-and-XOR mode has the limitation of being able to operate solely when the generation has exactly one full-length packet. To address this condition, we propose to combine it with the systematic mode. Practically, it is clear that the shift-and-XOR mode should only be used if there is no more than one full-length packet in the generation. Otherwise, the encoder operates on the systematic mode and communicates the first L_{max} macro-symbols systematically in one coded packet. Overall, if a generation of size N packets has η full-length packets ($\eta \in [\![1 \; ; \; N]\!]$), then at least $\eta - 1$ coded packets should be created using the systematic mode, in order to be able to recover the generation at the receiver. The source macro-symbols that are transmitted in a systematic fashion are expected to bring more degrees of freedom for full-length packets. This would enable the recovery of the remaining macro-symbols, which could not be solved by the shift-and-XOR mode on its own.

5.3.2.3 Macro-Symbol Policies for Full-Length Packets

Only for convenience, a generation that contains only full-length packets is better transmitted using the systematic mode. This approach is expected to outperform the systematic network coding when the generation contains variable-size packets since macro-symbols themselves reduce the padding overhead regardless of the macro-symbol scheme adopted. Nonetheless, the systematic mode becomes slightly costly in terms of computational complexity due to the higher number of macro-symbols when the packets are all full-length. This is because encoding and decoding operations are both performed on a practically larger generation, i.e., larger decoding matrices.

5.3.3 Impact of Packet Losses

As the RaSOR scheme cannot perform recoding, we propose some practical mechanisms for reliable point-to-point communications when losses occur. We assume the transmission of a generation of coded packets to a receiver via an erasure channel, which is capable of carrying one packet per time slot but has a loss probability ε. We also assume that packet losses are inter-independent across the channel and feedback is not permitted, otherwise, we would not fully benefit from the coding scheme's rateless feature. We note that different transmission strategies play a role in reducing the impact of losses on the solving of generations. They could be characterized based on the coding mode and the number of full-length packets in a generation. In the latter case, the sender could randomly shift-and-XOR the macro-symbols and transmit the resulting coded packets in a rateless fashion to guarantee to decode. Nonetheless, it could adopt the systematic mode first in order to transmit $\eta \leqslant \Delta_{\text{opt}} \leqslant N$ packets before switching to the random shift-and-XOR mode as proposed in [186]. In this manner, we assure the transmission of a maximum number of macro-symbols systematically, i.e. without investing in the shifting and the XOR computations, before moving to the non-systematic mode that leverages the recovery of lost macro-symbols without the need for Automatic Repeat reQuest (ARQ). Under the constraint that a generation contains a large number of full-length packets, the aforementioned strategy is certainly not accurate, as the systematic mode is not rateless. This means that full-length packets are not necessarily decoded. Consequently, solving this type of generation is impossible. To be more concise, in the following we emphasize the transmission of generations with one full-length packet only and leave the other case for future work.

5.3.4 Empirical Results

We propose to demonstrate the enhanced performance of the RaSOR scheme compared to the conventional RLNC and provide numerical results throughout extensive simulations since we were not able to determine analytically some performance metrics like the decoding delay. We start with the example in Figure 4.1, and show the rank evolution until complete decoding of the generation using a Markov chain, and additional simulations to have the exact numbers for the probability of decoding, as we showed with the other MS RLNC. Furthermore, we perform simulations on generations formed by the video trace from CIF [154].

As far as we know, there is no straightforward decoding probability formulation, which could express the expected number of coded packets needed for decoding. We also showcase that there is no generalized Markov chain, which could cover the rank evolution of the decoding matrix during the decoding process. We were actually constrained by the difficulty of characterizing the random behavior of the shifts, as well as the size distributions altogether. Nonetheless, we are constantly working on finding other analytical means that would characterize the decoding delay for a wider range of cases.

5.3.4.1 Basic Example

Figure 5.14 illustrates the rank evolution of the generation in Figure 4.1 at the receiver. The maximum rank to be achieved is $\sum_{i=1}^{4} \Lambda_i = 9$. We denote by p_i the probability of remaining at rank $rk = i$ when a received coded packet is not innovative, and p_{ij} represents the probability of the rank increase from being $rk = i$ to $rk = j$, $j \geqslant i$ when a received coded packet is innovative. It is clear that this graph changes completely based

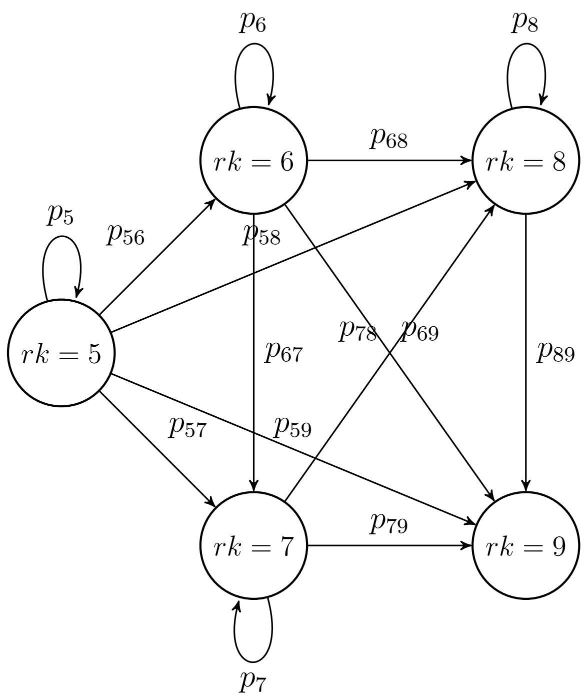


Figure 5.14: Markov chain for the generation in Figure 4.1. p_i represents the probability when the next packet is not innovative, and p_{ij} the probability of increasing the rank from i to j during one coded packet transmission, with $i \in [\![\Lambda_{\max} \; ; \; N-1]\!]$ and $j \in [\![\Lambda_{\max} \; ; \; N]\!]$, $j \leqslant i$.

on the generation in question, and it adapts to new possible achievable ranks. In general, we note that the first coded packet brings a rank $rk = \Lambda_{\max}$ since we cannot have a zero vector with RaSOR. The next coded packet P_i, $i = 1, 2, \cdots, K$, could bring multiple possibilities for the rank evolution as shown in the example of Figure 5.14.

We characterize this rank evolution using Figure 5.15, which depicts the probability of decoding the generation of $N = 4$ in Figure 4.1, in terms of the number of coded packets transmitted (K). We observe that the RaSOR scheme starts decoding at the theoretical optimal number $\Delta_{\text{opt}} = 2$ with a probability $\mathbb{P}^{(2)} = 0.173$. The conventional RLNC starts decoding at $K = N = 4$ coded packets, with a probability for GF(2) around 0.33, whereas RaSOR reaches a decoding probability of 0.96. Furthermore, the mean decoding probability of the RaSOR scheme for this example is $\mathbb{E}(K)_{\text{RaSOR}} = 3.0575$, which is by far more impressive than the mean value expected by RLNC for any finite field in use, without forgetting the fact that such a scheme has a reasonable encoding and decoding complexity thanks to its basic calculations that are limited to shifting and XORing the macro-symbols.

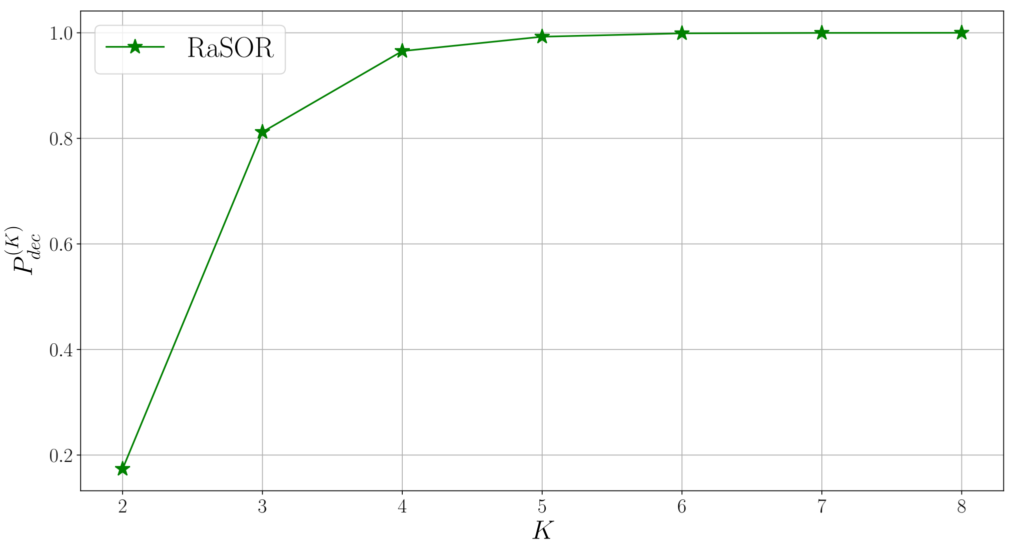


Figure 5.15: Probability of decoding using different schemes for the example in Figure 5.2 consisting of $N = 4$, $(\Lambda_1, \Lambda_2, \Lambda_3, \Lambda_4) = (1, 5, 1, 2)$.

5.3.4.2 Video Frames

CIF is one of the common resolution formats used for video surveillance by constrained devices. It is therefore important to employ a coding scheme that has low encoding and decoding computational complexities, like what RaSOR offers. We present the performance of the RaSOR scheme along with the conventional RLNC when applied to 32×10^3 of the SVC (CIF) video trace from the Arizona State University online library. Since RaSOR is constrained with the condition that a generation could be decoded only when it contains only one full-length packet, we propose to also employ the hybrid and systematic encoding modes. We present the expected number of coded packets needed for two scenarios, namely, the one without losses in order to emphasize the capabilities of the RaSOR scheme, and the one that considers losses to point out that RaSOR is also a rateless code.

Without losses Figure 5.16 depicts the average number of coded packets needed to decode as a function of the generation size for the hybrid encoding mode of the RaSOR scheme, conventional RLNC for GF(2) and GF(2^8), compared to the theoretical optimum case that recalculates the number of packets needed (Δ_{opt}) if we use macro-symbols. We observe at first glance an impressive decrease in K, the number of coded packets needed

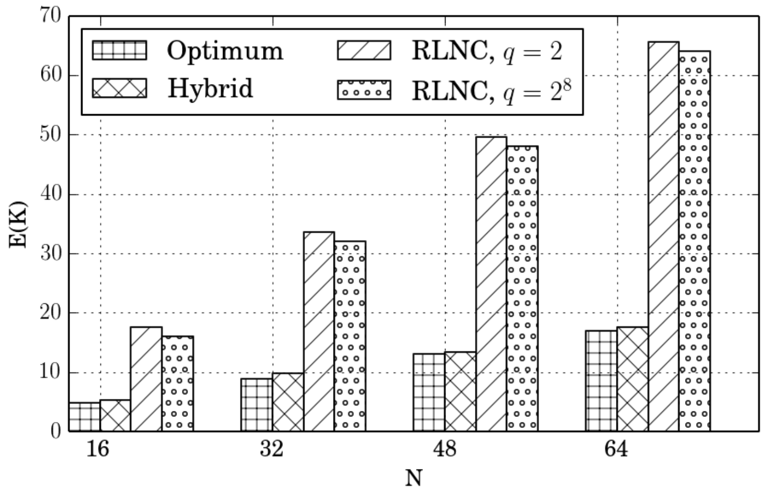


Figure 5.16: Number of coded packets needed in average when using various schemes to decode 32×10^3 packets of a sequence of the CIF video trace. N is the generation size.

for decoding compared with the original number N. For instance, we transmit around four-folds fewer coded packets with the hybrid mode compared to RLNC with $N = 64$. Furthermore, the overall performance of the hybrid mode is close to the theoretical numbers. This proves that extensively shifting the packets before encoding and sending is an effective approach in creating innovative packets with an overwhelming probability.

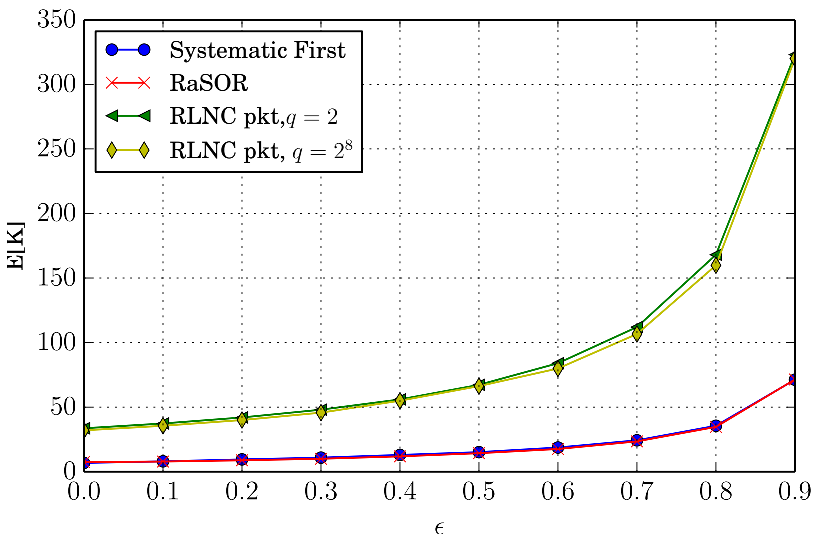


Figure 5.17: Number of coded packets sent on average in the presence of losses with probability ϵ, for $N = 32$, $\mu = 30$, 32×10^3 packets were used from the CIF video trace.

With losses Figure 5.17 illustrates the average number of coded packets needed when transmitting the portion of the video trace using generations of sizes $N = 32$ as a function of the loss rate ϵ. Expectedly, the number of coded packets K grows when ϵ increases. In comparison to the results from Figure 5.16, the difference in K between the RaSOR scheme and the conventional RLNC is maintained for all values of the loss probability ϵ. Nonetheless, when we adopt the systematic mode, we send one packet less on average compared to RaSOR. This has the potential to further improving the RaSOR scheme. We note that the results obtained with the conventional RLNC for GF(2^8) are similar to the results when we exclusively employ the systematic mode, because the number of original packets is always equal to N.

5.4 Conclusions

This chapter dealt with the design and implementation of macro-symbol-based techniques that rely on the pre-coding step of shifting the original packets before encoding. For instance, we proposed the *deterministic shifting*, *random shifting*, and *RaSOR* schemes that take advantage of the variable-size input packets to produce less coded packets instead of naively filling the shorter packets with zeros, and generate as many coded packets as the original ones. The extensive performance evaluation of macro-symbol approaches has shown that they are heavily dependent on the packet size distributions. The first couple of schemes fully inherit the standard RLNC features, e.g., recoding ability and drawbacks (like field choice dependency). The deterministic shifting scheme generates the least number of coded packets as in Eq. 5.1.

Moreover, the simple but effective pre-coding design provides an extra lightweight cipher for the data to disseminate, since, aside from coding on a macro-symbol basis, all packets are susceptible to being shifted.

However, RaSOR does not employ any coding coefficients, as it simply relies on shifting the packets extensively each time a new coded packet is to be created. In fact, we do not consider it as an RLNC based approach, since it does not have the recoding ability. It is also not rateless, since the number of possible combinations for coded packets is limited, and it is exclusively determined by the number of random shifting combinations within the set of unequal-sized packets. Moreover, we proved that the packet sizes play a crucial role in determining the effectiveness of this scheme, and proposed some simple mechanisms to counter this issue. Despite this limitation, it remains the cheapest and easiest approach to encode and decode unequal packets, with basically an expected padding overhead determined by the size μ of the macro-symbols. Finally, we believe that RaSOR could be oriented to be an instantly decodable code [40]. Furthermore, we highlight that this scheme allows for online coding [185], i.e., coding decisions are adapted according to the data arrival in the sender's buffer. This is an interesting feature for live video streaming applications that are extremely sensitive to delays.

Finally, the degree distribution of encoded macro-symbols during the transmission of the whole generation makes decoding using the simple Belief Propagation and inactivation decoding algorithms as for the BATS codes [60] possible, since both algorithms are known to have lower computational complexity compared to the Gaussian elimination.

Chapter 6 | Joint Compressed Sensing and Network Coding

Network coding and compressed sensing share a broad interest in various disciplines over the last decade. As they are currently considered to be a part of the core of 5G technologies, the idea of combining both techniques in order to maximize the systems' performance became very appealing. This translates into leveraging the benefits from network coding while exploiting the correlations in the data thanks to compressed sensing. The potential gains, including lower latency for large-scale sensing scenarios, lower energy consumption, and a decrease in the amount of data during transmissions, are alluring to many use-cases. However, efficiently joining the techniques requires a careful tweak in their designs to meet the expected improvements compared to an agnostic combination. Network coding was mainly designed to operate on finite fields, whereas compressed sensing is primarily concerned with real numbers.

This chapter focuses on highlighting the common characteristics and the requirements for an effective combination approach, through understanding and determining the right field over which to operate, the coding and compression matrices design, as well as the efficient algorithms for a fast and accurate reconstruction. To do so, we propose the design of Joint Compress and Code (JoComCo) scheme for cluster-based WSNs that enables a one-step decoding strategy, instead of agnostically cascading the reversion operations from both techniques aside. It consists of exploiting the temporal and spatial correlations among the sensory data, as well as intra-flow and inter-flow real network coding exploiting the cluster topology. We employ the KL1p compressed sensing library and the Network Simulator 3 (ns-3) [187] to evaluate this joint design. Our results show that normalized coefficients from Gaussian distributions provide higher scalability and efficiency for multi-hop networks, where the recoding feature of network coding is required.

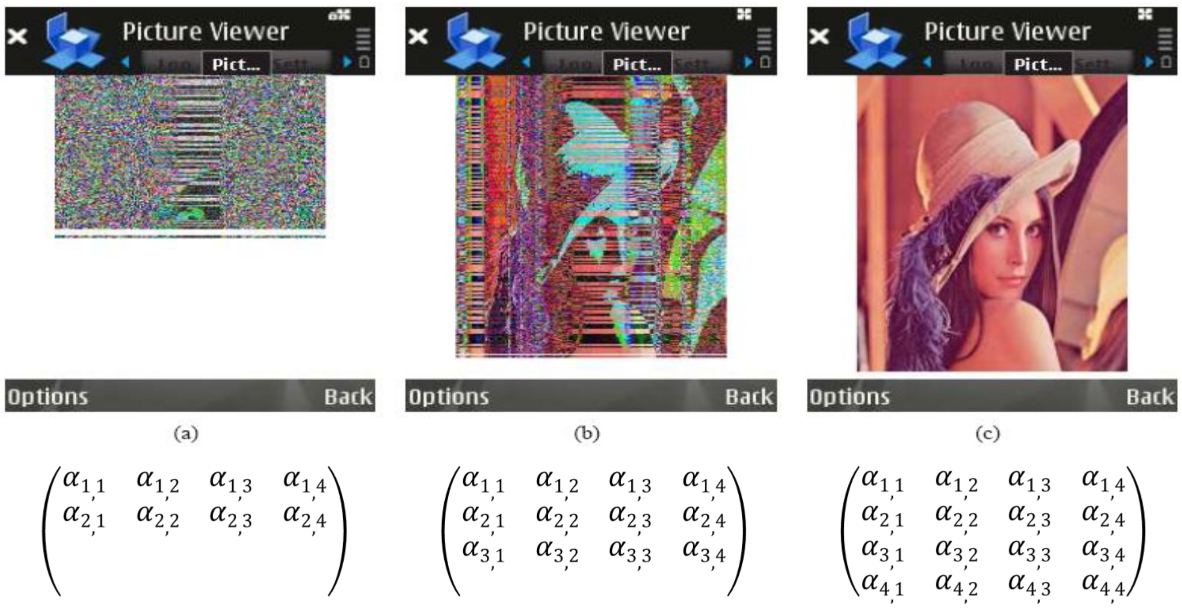


Figure 6.1: On the fly decoding of Lena's picture originally encoded using GF(2); (a) the data partially is decoded, (b) the image starts to appear as the decoding matrix rank increases, (c) the image is completely decoded [7].

Additionally, the SP reconstruction algorithm outperforms the benchmark algorithms provided by the *KL1P* library [14], with respect to the Reconstruction Signal-to-Noise Ratio (RSNR), by more than two folds. Moreover, our scheme dramatically reduces the number of transmissions and their payloads by around 94%, while maintaining a high RSNR. We note that the content of the remaining of the chapter was published in [8–10].

6.1 SoA on the Combination of Compressed Sensing and Network Coding

The idea behind a potential compression gain from RLNC relates to 2009 when Pedersen et al. [188] observed that having less number of coded packets required to decode a generation in GF(2) did not prevent them from already guessing the original generation, which has led them to think of an inherent compression capability that could be further investigated. Figure 6.1 depicts the decoding progress of the famous Lena picture that was originally encoded using GF(2). Being able to observe early guesswork returns to the

Table 6.1: Compressed sensing and network coding common characteristics.

Compressed Sensing	Network Coding
Linear superposition	Linear superposition
Random (sampling)	Random (coefficients)
Source aware	Source agnostic
under-determined	Over-determined (full rank)
optimization problem	Linear system of equations
$\mathbf{y}^{m\times 1} = \mathbf{A}^{m\times n}\mathbf{x}^{n\times 1};\ m << n$	$\mathbf{y}^{(n+r)\times 1} = \mathbf{A}^{(n+r)\times n}\mathbf{x}^{n\times 1};\ r \geq 0$

fact that the decoding matrix is sparse, as these results are not observable when larger finite fields are in use. The idea is therefore to investigate the common characteristics that could relate to network coding and compressed sensing, as well as the requirements for a cross potential.

6.1.1 Common Characteristics and Requirements

The main characteristics of compressed sensing and network coding are both summarized in Table 6.1. The fundamental differences reside in the fact that compressed sensing is concerned with under-determined linear systems of equations, i.e., it relies on optimization algorithms to solve the problem and finding exact or approximate solutions, whereas network coding cannot solve the linear system of equations without acquiring at least as many coded packets (equations) as the original packets (unknowns), i.e., the *all-or-nothing* condition, leading to using the Gaussian elimination algorithm to retrieve the original data. Furthermore, compressed sensing is aware of the data characteristics, i.e., the sparsity or compressibility, and employs this knowledge in the reconstruction procedure. However, network coding is completely agnostic to the data it operates on.

Performing network coding in large-scale WSNs guarantees the enhancement of the overall network performance thanks to its impressive features that we discussed in Chapter 2, such as exploiting the path diversity, error correction, and the broadcast nature of WSNs. Unfortunately, RLNC's successful decoding relies on acquiring as many linearly independent coded packets as the original ones. Compared with the most promising ability of compressed sensing, this is a critical drawback that will hold RLNC back from participating in reducing the number of transmissions despite the spatial correlations that are inevitable in WSNs. Furthermore, applying intra and inter nodes distributed compressed sensing [189] suppresses this constraint, but cannot be as robust as network coding against erasures.

On the other hand, the idea of agnostically combining compressed sensing and network coding could be more efficient, in the sense that together they guarantee a decrease in the payload and the number of packets transmitted even in the presence of erasures. Nevertheless, the data reconstruction procedure cannot be faster because it simply requires several independent decoding steps, to revert the agnostic operations of network coding and compressed sensing. Consequently, both techniques should operate using the same field type, as well as having compatible coding and compression matrices, to obtain the full effectiveness of a joint approach. As far as we know, these are the most important characteristics to guarantee one-step joint decoding. Subsequently, it is expected to have lower energy consumption and lower computational complexities.

It is, therefore, crucial to understand the cross variants of compressed sensing and network coding, in order to be able to choose the expected beneficial option that does not agnostically combine the techniques, but rather modify them in a way that guarantees higher gains. For instance, we discuss in the following the branches of real network coding and finite field compressed sensing.

6.1.1.1 Real Network Coding

The traditional RLNC implementation favors algebraic operations over finite fields instead of the real field as these operations are expected to generate larger results than the allocated space inside the coded packet, leading the coded packet and encoding vector sizes to explode. Additionally, the recoding and decoding procedures increase in complexity as the coded packets do not have matching sizes any more. These RLNC features are not guaranteed when operating on other than finite fields and might result in reconstruction inaccuracies or errors. On the other hand, the problem of compressed sensing is mainly concerned with recovering a certain sparse signal using only far few samples. Such a property does not go hand in hand with the *all-or-nothing* property of finite field network coding. Moreover, finite fields do not provide normalization for the finite field vector spaces, as simply there are no such things as the ℓ_p ($p \geqslant 1$) norms in finite fields.

Consequently, it seems to be more beneficial to realize a combination approach when both techniques operate in the real field, in order to improve the expected gain. Accordingly, real network coding is designed in a way that complies with the compressed sensing [190], i.e., the coding matrix must satisfy the sensing matrix properties, e.g., RIP, Restricted Eigenvalue (RE), etc., or simply drawn from a Gaussian distribution. Therefore, it is currently mandatory to think of a real field version of network coding, which also preserves the fundamental features of RLNC. The idea of real network coding was introduced in 2007 slightly after the advocacy of the compressed sensing field [190]. Additionally, it is reasonable to think of a real network code as signals in practice are better described in the real domain. This inspiration aims mainly to break with the *all-or-nothing* principle while mimicking the compressed sensing behavior.

The difference relies on the fact that real network coding remains unaware of the sparsity present in the data, i.e., it does not rely on the data correlations. Therefore, this class of network codes benefits from features of both areas, providing an attractive potential in increasing the network's throughput and reliability, while decreasing the computational complexity, and consequently the delays. The techniques combined are expected to deliver progressively more accurate approximations of the data, unlike the *all-or-nothing* property. Nevertheless, this combination comes at the cost of some practical challenges that we will further discuss in this chapter, including finite precision during decoding, as well as the vector overhead explosion during the encoding process. For this reason, it is essential to study the impact of the coding matrix distributions on the reconstruction accuracy [9, 10, 191].

6.1.1.2 Finite Field Compressed Sensing

Draper et al. [192] were the first to introduce compressed sensing in finite fields that mainly targets sparse finite-alphabet sources. They developed an error exponent theoretical framework of sparse finite alphabet sources in the absence of noise while parameterizing the alphabet size as well as the level of sparsity. Based on their results, Seong et al. [193, 194] proposed some necessary and sufficient conditions for the recovery of such sparse signals, in terms of the dimension of the signal space, the sparsity, and the number of measurements. They proved that the sensing matrix should be dense to ensure the recovery of very sparse signals. As the RIP is hard to verify for finite field matrix because ℓ_2 cannot be accurately computed for finite fields, the NSP is a more reliable option [192].

Interestingly, perfect reconstruction could be achieved using a number of measurements that is larger than the support for sufficiently large and dense matrices in a large finite field size [193]. Furthermore, only two other works were devoted to security [195], as well as sparse image recovery using a Finite Field Orthogonal Matching Pursuit (F2OMP) reconstruction algorithm [196]. The research related to finite field compressed sensing remains a challenging topic, that requires tighter properties than the ones proposed for standard compressed sensing.

6.1.2 Combination Results of SoA Approaches

With a strong emphasis on WSNs and IoT in general, there exist several research works that considered the combination of these two innovation techniques in different areas, including but not limited to wireless communications, security, and data privacy [197], spectrum sensing [198], distributed stoarge [199], etc. For instance, Nabaee et al. [200,201] proposed a quantized network coding and compressed sensing approach for correlated sources in order to reduce the delivery delays of the network thanks to the few transmissions needed at the destination for reconstruction. The idea of compressed sensing and analog network coding, which is a physical-layer network coding [202], was proposed for WSNs chain-type topologies to effectively reduce the energy consumed by the sensors, thus increasing their lifetime [203, 204].

Most of the efforts focused on the combination design with the real field, because the compressed sensing properties are tighter than that of the conventional RLNC [205, 206], as previously discussed. Nevertheless, we simultaneously noticed that only a little attention was given to the temporal and spatial correlations. For instance, Nguyen et al. [207] proposed the *Netcompress* approach for WSNs that enables the sensors to compress their data before applying real network coding at the intermediate nodes. Unfortunately, the scheme could not prevent packet header explosion [208], which has led to dropping some of the packets whenever this issue occurs. This has resulted in a limited compression gain for the payload size, partially due to the temporal compressed sensing. Furthermore, the works by Feizi et al. [209, 210] exploited the cross potential between compressed sensing and real network coding and discussed interesting theoretical gains, but without further implementation for validation. *Compressive Network Coding* is a joint source and network coding method for approximate data gathering that is robust against burst transmission errors and maintains a good overall performance when the values of the sensor readings abruptly change [211]. Finally, Chen et al. [204] provide a table that summarizes the characteristics and drawbacks of the common state-of-the-art approaches in.

Other works focused on proposing a finite field compressed sensing approach in order to maintain the salient features of conventional RLNC [192–194,196,212]. These combined techniques aim to also solve the *all-or-nothing* issue of RLNC that could hinder the benefits from compressed sensing [213]. For example, Xie et al. [197] proposed a privacy-preserving approach that secures network-coded packets using finite field compressed sensing at the source. The framework can resist data and tag pollution attacks. Moreover, common compressed sensing influenced algorithm used for this type of combination is the finite field OMP [214], sometimes referred to as F2OMP [196].

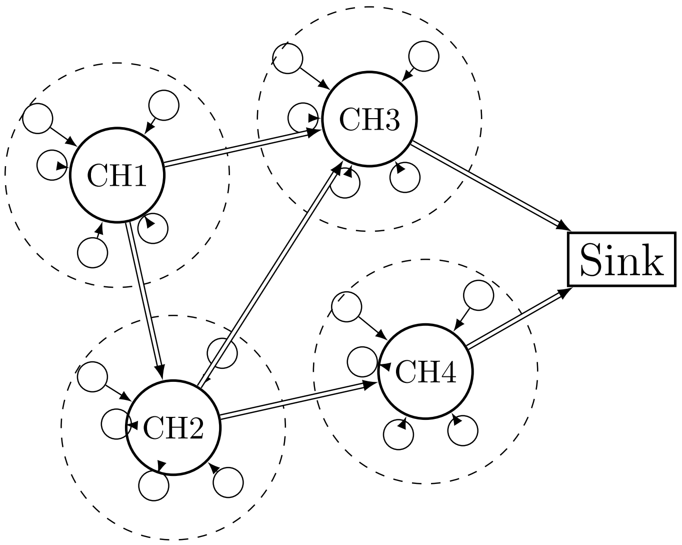


Figure 6.2: An example of a four-cluster topology [8], where CHi, $i = 1, \cdots, 4$ is the cluster head.

Additionally, Rajawat et al. [215] focused on formulating a maximum a posteriori estimation for the Bayesian finite field compressed sensing and proposed a low-complexity sum-product algorithm for factor graphs. A similar idea was proposed by Wenjie et al. [101, 216]. Nevertheless, we believe that finite field compressed sensing loses a lot of its features and sometimes could become impractical due to the constrained finite field design. Therefore, investing in the research of new approaches that reduce the computational complexities and obey the basic compressed sensing properties, mainly NSP, could be an interesting future research direction.

6.2 Joint Compress and Code (JoComCo) Scheme

In order to overcome the drawbacks of the state-of-the-art approaches, we propose the design of an in-network computing scheme, which does not only exploit the temporal and the spatial correlations of the sensors and the corresponding data but also allows for the fusion of network coding and compressed sensing while avoiding the drawbacks of the previously discussed agnostic approaches. To do so, we introduce in this section the design of the JoComCo scheme, which relies on real network coding and conventional compressed sensing. This approach is designed for cluster-based WSNs topologies.

6.2.1 Design for a Cluster-Based Topology

To better illustrate the problem and propose a related solution, we propose a simple topology consisting of four-cluster of N sensors, including cluster heads (CH$_j$), $j = 1, \cdots 4$, and a sink, as illustrated in Figure 6.2. A sensor node is an entity capable of connecting (wirelessly) to its related CH – including the cluster head itself – and can harvest and transmit specific measurement values. Moreover, a node should have enough available resources to execute the necessary algorithms. The cluster heads are provided with more hardware resources and higher computational capabilities and are the only nodes in charge of communicating with one another, as well as the sink.

6.2.1.1 Temporal Compression

We assume that the sensors are all capable of sampling their readings and that they are observing the same natural phenomenon or process.

Let $\nu_k \in \{\nu_1, \nu_2, ..., \nu_N\}$ be one of the sensor nodes and $\mathbf{x_i} \in \mathbb{R}^n$, $\mathbf{x_i} = [x_{i1}, \cdots, x_{in}]$, the readings vector obtained by ν_k in n time slots. The sensory data are supposed to follow the JSM-2 model [148], known as the common sparse support, in which case the signals are constructed from the same basis vectors while having different coefficients as follows:

$$\mathbf{x_i} = \mathbf{\Psi_T}\theta_\mathbf{i}, \quad i \in \{1, \cdots, N\}, \tag{6.1}$$

where $\mathbf{\Psi_T} \in \mathbb{R}^{n\times n}$ is the sparsification matrix of $\mathbf{x_i}$, and $\|\theta\|_0 = k$. Let $\mathbf{X} = [\mathbf{x_1}, \cdots, \mathbf{x_N}]^T$ and $\mathbf{\Theta} = [\theta_\mathbf{1}, \cdots, \theta_\mathbf{N}]^T$ be the matrices for the N sensed vectors and their representations in the sparse basis, respectively.

Every sensor $i = 1, 2, \cdots N_j$ belonging to a cluster $j = 1, 2, \cdots$ performs a temporal compression on its readings $\mathbf{x_{ji}}$ that results in a dimension reduction of the resulting measurement $\mathbf{y_{ji}}$ as follows:

$$\mathbf{y_{ji}} = \mathbf{\Phi}\mathbf{x_{ji}}, \tag{6.2}$$

where $\mathbf{\Phi} \in \mathbb{R}^{m\times n}$ represents the measurements matrix that obeys the RE property [209, 217]. This matrix is applied similarly to all sensor readings and should be known by the sink in order to be able to effectively reconstruct all the data. We denote the matrix containing all the temporally compressed vectors by $\mathbf{Y_j} \in \mathbb{R}^{m\times N_j}$, as $\mathbf{Y_j} = [\mathbf{y_{j1}}\mathbf{y_{j2}}\cdots\mathbf{y_{jn}}]^T$.

6.2.1.2 Spatial Compression

The spatial compression (also called spatial pre-coding) consists of randomly selecting the measurements that are kept for further communication steps. We recall that sensors are generally geographically close to one another, leading to a high probability of generating correlated data. As such, every sensor decides on its transmission based on the distributed probability among all the sensors involved.

The final selected data, denoted by $\mathbf{Z_j}$, to be transmitted by every sensor j to the corresponding cluster head is defined as:

$$\mathbf{Z_j} = \mathbf{B_j}\mathbf{Y_j}, \tag{6.3}$$

where $\mathbf{B_j} \in \mathbb{R}^{a_j \times N_j}$ is the pre-coding matrix, and a_j is defined based on the transmission probability p_j. Furthermore, we define $\mathbf{B} \in \mathbb{R}^{N \times N}$ as the diagonal matrix containing $\mathbf{B_j}$, $j \in \{1, \cdots, N\}$. As such, $\mathbf{B_j}$ are diagonal sub-matrices, expressed as $\mathbf{B_j} = diag((b_{ji})_{i \leqslant N_j}$. According to Feizi et al. [210], the pre-coding is better performed using the simple Bernoulli distribution if the matrix $\mathbf{\Phi}$ obeys the RE property. However, we suggest introducing some modifications based on real network topology requirements and constraints, e.g., loss rates. A sensor is allowed to transmit its measurement to its corresponding cluster head based on the following distribution:

$$p(b_{ji} = 1) = 1 - p(b_{ji} = 0) = \varrho_{ji}\mu_j \frac{l_j - 1}{N_j - 1}, \tag{6.4}$$

where the modifier ϱ_{ji} gives support to $\mathbf{y_{ji}}$ in coping with link losses of probability ε_{ji}, as follows:

$$\varrho_{ji} = \frac{1}{1 - \varepsilon_{ji}}. \tag{6.5}$$

As for μ_j, it basically depends on N_j and the number of measurements expected to be received by the cluster head, l_j, and its selection is developed in Appendix A. Based on our measurement campaign reported in the following section, we will fix the convenient value of μ_j for our numerical results.

"On the other hand, since the sensor nodes independently decide on their own transmissions to the cluster head, the modifier enables CH_j of receiving $(l_j - 1)$ measurement vectors with a high probability P_{rx}, defined by

$$\mathbb{P}(r \geqslant l_j - 1) = \sum_{i=0}^{l_j-2} \binom{N_j - 1}{i} P_{rx}^i (1 - P_{rx})^{N_j-1-i}, \tag{6.6}$$

where r denotes the number of received packets (see Appendix A for the detailed description and parameters selection). We firstly employ a diagonal Bernoulli matrix $\mathbf{B}$ for referring to the sensors [218], defined as follows :

$$b_{ii} = \begin{cases} 1, & \text{with probability } \mu_j \frac{l_j-1}{N-1} \\ 0, & \text{with probability } (1-\mu_j)\frac{l-1_j}{N_j-1}, \end{cases} \tag{6.7}$$

where b_{ii} are the diagonal elements and μ_j is the transmission probability modifier. μ_j guarantees that at least l_j data vectors – where l is the trace of $\mathbf{B}$, denoted by $tr(\mathbf{B})$ $(1 \leqslant l \leqslant N)$ – at the cluster head are received with a high probability. Consequently, the resulting spatial compressed sensing matrix $\mathbf{B} \in \mathbb{R}^{N\times N}$ is defined as $\mathbf{B} = \text{diag}((b_{ii})_{i\leqslant N})$" [8].

Given that the sensors are geographically close, they are also potentially spatially correlated. Therefore, some of the values in the diagonal sampling matrix $\mathbf{B}$ will be zeros with $(1-\mu_j)$ probability, meaning that not all of the sensors will be polled for their readings. This will potentially reduce the computation and transmission costs at these nodes – which decreases the overall resource requirements in general – and one will still be able to recover the original data with sufficiently high probability. This step can be further improved by using a minimal amount of data prediction at the cluster head. Such information is acquired from the cloud or the sink based on the history of the collected data, as per [219]. Our suggested framework relies on random selections among the sensors of a cluster. Therefore, it is possible that we are not getting the minimal amount of data from the sensors and the compression is not optimized. However, it is still a non-expensive way of collecting data from geographically close sensors. Finding an accurate collection technique, which guarantees, amongst other features, the minimization of the energy consumption, seems to be an NP-hard problem, and an exploration of this is not in the scope of this dissertation.

The $(N-1)$ sensors, excluding the cluster head *CH*, communicate their measurements to *CH*. This part of the compression is therefore considered as the *temporal compression* step, which is also low-complexity and is resolved locally on the sensor device.

6.2.1.3 Intra-Cluster Real Network Coding

Since compressed sensing on its own cannot ensure the reliable transmission of the readings from the m measurements needed at the sink for data recovery, we introduce a FEC coding scheme, specifically intra-cluster network coding to account for the recovery and transmission of the missing packets. After the spacial compression step, the cluster head performs a real network coding operation on the already compressed measurements as follows:

$$\mathbf{z_i} = \mathbf{\Omega} b_{ii} \mathbf{y_i}, \tag{6.8}$$

where $\mathbf{z_i} \in \mathbb{R}^m$, $\mathbf{\Omega} \in \mathbb{R}^{l \times N}$, $\mathbf{i} \in \{1, \cdots, l\}$, and $\mathbf{\Omega}$ is the network coding matrix, and its coefficient distribution will be further discussed. This in turn translates to:

$$\mathbf{Z} = \mathbf{\Omega} \begin{bmatrix} b_{11} & 0 & \cdots & 0 \\ 0 & b_{22} & \cdots & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & b_{NN} \end{bmatrix} \begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \dots & \mathbf{y}_N \end{bmatrix}^T = \mathbf{\Omega B Y}^T. \tag{6.9}$$

6.2.1.4 Inter-Cluster Real Network Coding

Once all cluster heads are done with the intra-cluster network coding stage, they start communicating their resulting data with one another following the random link connections depicted in Figure 6.2. When an intermediate cluster head collects the expected packets from other cluster heads, it recombines (recodes) all incoming and own packets, using a set of randomly generated coefficients belonging to the corresponding matrix $\mathbf{\Omega^r}$. Therefore, the entire network coding operations intra and inter-clusters can be captured by $\mathbf{\Omega\Omega^r}$. They are supposed to be designed such that they are jointly capable of conserving the stringent RE property. It is also worth mentioning that these real-valued random matrices have the full-rank property with a high probability for a sufficiently large a_i. Additionally, the network coding operations at the cluster heads are expected to reduce the total number of redundant transmissions as well as compensating for the lost ones thanks to its effectiveness as an FEC. Consequently, we propose to convey an extra number of coded transmissions by the cluster heads to cope with the predefined packet loss probability.

Let $\mathbf{U} \in \mathbb{R}^{l \times m}$ be the matrix that represents all coded and compressed data transmitted by the cluster head to the sink. We express the overall aforementioned procedure as:

$$\mathbf{U} = \mathbf{\Omega^r \Omega B Y}. \tag{6.10}$$

6.2.2 Reconstruction at the Sink

We recall that the temporal and spatial compression, as well as the network coding steps, can be summarized as follows:

$$\begin{bmatrix} \mathbf{z_1} \\ \mathbf{z_2} \\ \vdots \\ \mathbf{z_l} \end{bmatrix} = \underbrace{\mathbf{\Omega B} \begin{bmatrix} \mathbf{\Phi} & 0 & \cdots & 0 \\ 0 & \mathbf{\Phi} & \cdots & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \mathbf{\Phi} \end{bmatrix} \begin{bmatrix} \mathbf{\Psi_T} & 0 & \cdots & 0 \\ 0 & \mathbf{\Psi_T} & \cdots & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \mathbf{\Psi_T} \end{bmatrix}}_{\mathbf{\Gamma}} \begin{bmatrix} \theta_1 & \theta_2 & \dots & \theta_N \end{bmatrix}^T, \tag{6.11}$$

where $\mathbf{\Gamma}$ is the matrix that combines all coding steps performed before the cluster head transmits the resulting packets. Assuming that r packets ($r \leqslant l$) are received by the sink, our goal is the reconstruction of the original data with high accuracy using the aforementioned coding matrices. The joint decoding is ensured using the common compressed sensing reconstruction algorithms. Note that for the reduction on the constraints in our design, we assume that the sink is aware of the sparsifying matrix, the measurement matrix, as well as the network coding matrix coefficients by the means of a shared seed or through the encoding vectors of the received packets.

6.3 Matrices Distributions and Reconstruction Algorithms

In this section, we first introduce and take a closer look at the joint sensing matrices and the available reconstruction algorithms. The main questions that we propose to answer in the remaining of this chapter are:

i) What are the properties of the matrices for both techniques, and which should be satisfied to enable the seamless combination of network coding and compressed sensing, while preserving the maximum number of features?

ii) Which kind of recovery algorithms are applicable for such an approach, and which are the most relevant ones?

6.3.1 Joint Sensing and Coding Matrices

According to compressed sensing literature, producing adequate matrices that ensure a high reconstruction accuracy is an intriguing and complicated endeavor. The construction of such matrices in a way that they are optimal for compressed sensing remains an open problem. Nevertheless, a breakthrough is achieved when using random matrices. The most popular distributions are the *Gaussian*, *Bernoulli*, *Rademacher* and *subgaussian* distributions. They could allow the reconstruction from Eq. 2.12 with a sufficiently high probability using a variety of algorithms, which we will discuss later on. As for network coding matrices, we are no longer bound by the requirement of finding a non-singular matrix, however, one needs to acquire a matrix that is as close to a non-singular one as possible, while retaining the characteristics of a compressed sensing matrix. This is proved by the *Cascading lemma* in [209].

Bernoulli Distribution "The interesting feature of the discrete probability distributions – mainly the Bernoulli distribution – is that the sensing (coding) coefficients need only 2 bits of storage. They are easy to implement in the sense that the packets are either added or subtracted randomly. Additionally, by the efficient storage of coefficients, the explosion of encoding vectors is avoided. Nevertheless, it gets problematic to have two or more matrices drawn i.i.d. from the Bernoulli distribution during multi-hop transmissions, where the recoding feature of network coding is crucial as the resulting distribution is not Bernoulli, but rather subgaussian" [10].

Gaussian Distribution "The multiplication of several Gaussian matrices will result in a Gaussian matrix, thus the *RE property* is preserved and the reconstruction is possible. Additionally, in order to avoid the problem of header explosion – which is obvious in the multi-hop scenario –, we propose a normalization of the coefficients. Nonetheless, Gaussian matrices are more power-consuming for the sensors compared to the Bernoulli matrices, which do not require much storage space. Moreover, their coefficients could be translated into boolean values, thus having less computational complexity during the compressed sensing step [220]" [10].

6.3.2 Reconstruction Algorithms

It is important to recognize that the decoding algorithms of standard network coding do not hold anymore, as we cannot fulfill the all-or-nothing requirement. In general, when dealing with under-determined linear systems of equations, the matrix inversion procedure of the Gaussian Elimination is not possible under these settings. For this reason, the common optimization algorithms adopted by compressed sensing can be fully exploited. To reduce the task of implementing these already available algorithms, we use a large set of reconstruction algorithms from the *KL1p* [14] library. Table 6.2 lists the algorithms that we propose to use along with their complexities and the order of the number of measurements they require for accurate reconstructions.

Table 6.2: Comparison of computational complexities for reconstruction algorithms in the *KL1p* library [9, 14].

Algorithms	Complexity	Measurements
BP	$\mathcal{O}(n^3)$	$\mathcal{O}(k \log(n))$
OMP	$\mathcal{O}(k\, m\, n)$	$\mathcal{O}(k \log(n))$
ROMP	$\mathcal{O}(k\, m\, n)$	$\mathcal{O}(k \log^2(n))$
CoSaMP	$\mathcal{O}(m\, n)$	$\mathcal{O}(k \log(n))$
SP	$\mathcal{O}(k\, m\, n)$	$\mathcal{O}(k \log(n/k))$
EMBP	$\mathcal{O}(n \log^2(n))$	$\mathcal{O}(k \log(n))$

6.3.3 Theoretical Compression Gain

The standard compressed sensing gain (temporal compression in our case) is basically defined by the ratio of the number of measurements m over the original data size n. This means that the gain in the payload for a packet $\mathbf{y_{ji}}$ is defined by ε_T as

$$\varepsilon_T = 1 - \frac{m}{n}, \tag{6.12}$$

which is quite high since $m << n$. This is expected to considerably reduce the payload size, thus the probability of erasures, which is desired in WSNs known to have high loss rates especially when placed in harsh environmental conditions, e.g., forests.

We define the overall compression gain by $\varepsilon_S \varepsilon_T$, where ε_S is the gain in the number of total transmissions in the network due to spatial compression and ε_T is the gain in payload size of packets due to temporal compression. Let us consider the single cluster case first. With only spatial compression, the compression gain is $\frac{N-1+l}{2N-1}$, where l is the number of packets sent from the cluster head to the sink for the reconstruction. The value of l is selected to ensure that the required RSNR is achieved with high probability. The compression gain achieved by joint CS and NC scheme is:

$$g^{(1)}_{JoComCo} = \frac{N-1+l}{2N-1}\frac{m}{n}. \tag{6.13}$$

6.4 Implementation Results

In this section we evaluate the performance of the network topology in Figure 6.2 using the NS3 simulator [187], along with the KL1p [14] library. We are interested in investigating the topology parameters that guarantee high compression gains, e.g., RSNR, average time for reconstruction, reconstruction probability, etc. Furthermore, we determine based on our simulation results the most convenient reconstruction algorithms, as well as the most efficient measurement and coding matrices.

6.4.1 Evaluation Setup

In the following, we define our performance metrics to determine the efficiency of the JoComCo scheme, as well as the right modifier μ introduced to improve the pre-coding procedure, which should be the first step in our measurements campaign.

6.4.1.1 Performance Metrics

The sparse data model considered for our simulation campaign is artificially created due to the lack of available large data sets at the time we conducted these measurements. This enabled us to smoothly tune the number of sensors, the number of readings, as well as the degree of sparsity k. We assumed that the sensory data $\mathbf{x_i}$, $i \in \{1, \cdots, N\}$ can be sparse in the DCT domain in such a way that $\mathbf{\Psi_T}$ is equal to the matrix of an inverse of the DCT. We note that all the sensors have the same temporal frequencies, since we adopt the JSM-2 model. For every sensor node i and each cluster head j, the coefficient α_{ij} is given by

$$\alpha_{ij} = A_j \cos\left[\pi F_j \left(i + \frac{1}{2}\right)\right], \quad F_j \in [0,1], \quad A_j > 0, \tag{6.14}$$

where F_j is an arbitrary discrete frequency. The amplitudes A_j are chosen randomly and are in turn normalized such that

$$\|\mathbf{X}\|_2^2 = 1 \quad \text{thus,} \quad \|\mathbf{x_i}\|_2^2 \approx 1, \quad \forall i = 1, 2, \cdots . \tag{6.15}$$

"In order to measure the performance of the decoded data, we propose the calculation of the ratio between $\|\mathbf{x_i}\|_2$ and the absolute error with the estimate $\|\mathbf{x_i} - \hat{\mathbf{x}}_\mathbf{i}\|_2$. Therefore, we define the Reconstruction SNR in dB for a sensor node i by, $RSNR_i$ as:

$$\mathrm{RSNR}_i = 20 \log_2 \left(\frac{\|\hat{\mathbf{x}}_\mathbf{i}\|_2}{\|\mathbf{x_i} - \hat{\mathbf{x}}_\mathbf{i}\|_2} \right). \tag{6.16}$$

Additionally, we compare to the non-noisy vector, as compressed sensing techniques are even capable of canceling noise due to the sparse selection. Analogous to Eq. 6.16, we define the recovery performance of the spatial decoder as:

$$\mathrm{RSNR}_S = 20 \log_2 \left(\frac{\|\mathbf{Y}\|_2}{\|\mathbf{Y} - \hat{\mathbf{Y}}\|_2} \right) \quad \mathrm{dB}. \tag{6.17}$$

However, we only have the ability to compare to the noisy $\mathbf{Y}$, since the simulation does not yield a noiseless version of it. Bearing in mind that in a real life scenario a processing application requires a minimum SNR, we define that the data for a source node was decoded properly if

$$\mathrm{RSNR}_i \geqslant \mathrm{RSNR}_{min}, \tag{6.18}$$

where RSNR_{min} is a configurable lower bound. Finally, for S_{meas} measurement sequences we can calculate the decoding probability as:

$$\mathbb{P}\left(\mathrm{RSNR}_i > \mathrm{RSNR}_{min}\right) = \frac{1}{S_{meas} N} \sum_{t=1}^{S_{meas}} \sum_{i=1}^{N} \left[\mathrm{RSNR}_{i,t} > \mathrm{RSNR}_{min}\right]. \tag{6.19}$$

We denote successful decoding when this probability is approximately equal to one" [10].

6.4.1.2 Selection of the Modifier μ

Figure 6.3 illustrates the optimal values for the modifier μ_j, for cluster sizes $N_j = \{64, 256\}$, and the number of measurements created $l_j = \{16, 32, 128\}$. It is naturally expected that the more measurements generated, the better the reconstruction is at the sink. For example, μ_j reaches approximately 1.2 when the network topology considered has $N_j = 256$ sensors per cluster and is capable of generating in total $l_j = 128$ measurements, and $\mu_j \approx 1.4$ if we only consider $l_j = 32$, which is not optimal. The value of μ_j is important to ensure an overall compression gain for the JoComCo scheme. In the remaining of this chapter, we fix $\mu_j = 1.2$

In the following, we evaluate the parameters for our proposed scheme and network topology and discuss our considerations. Then, we show the performance of the KL1p algorithms from the point of view of time and reconstruction accuracy. Finally, we

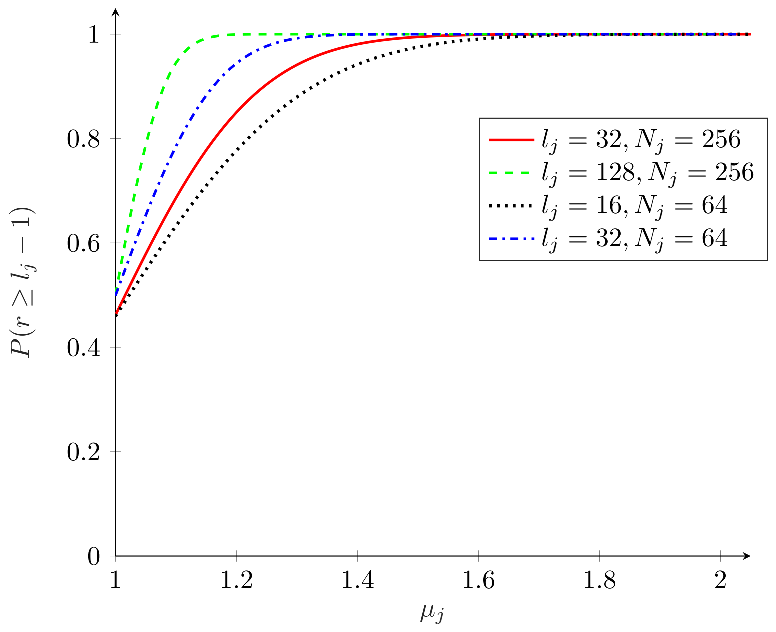


Figure 6.3: Probability to receive more than $(l_j - 1)$ measurement vectors at the CH_j, $j = 1, \cdots, 4$ [8].

evaluate our choices of the matrix distributions for a joint combination of compressed sensing and network coding.

6.4.2 Parameter Selection in a Single-Cluster Topology

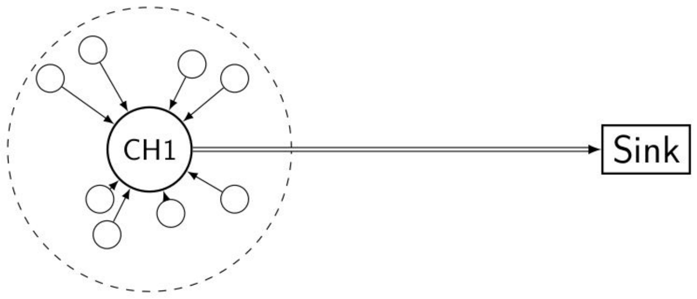


Figure 6.4: A single cluster topology with N sensors including the cluster head CH_1, which is a more powerful sensor compared to the rest of the cluster sensors, and a sink [9, 10].

To ease the constraints on our previously defined four-cluster topology, we opt for a simplified version consisting of one-cluster only as depicted in Figure 6.4. We believe that it is helpful in determining the right parameter selection for a generalized topology, and provides a good understanding of the behavior of compressed sensing and coding matrices.

First of all, we need to fix and validate the parameters m, for temporal compression, and l for spatial compression. We initially set the number of sensors involved in the simulations to $N = 256$ nodes. Also in order to show clear results, we only consider error-free links in our simulations and we disable spatial precoding, i.e., the transmission probability p_{tx} of the common sensor nodes is one. For these settings, the l number of packets needed at the sink for spatial reconstruction corresponds to the r number of packets that the receiver actually sees.

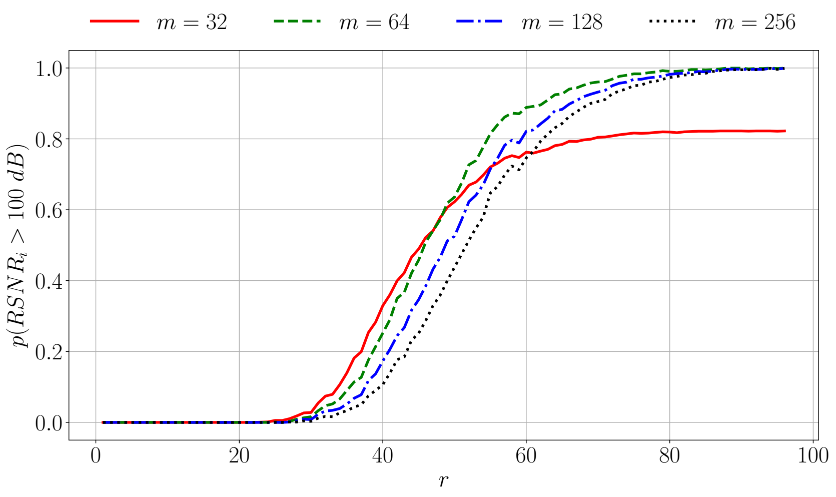


Figure 6.5: CDF of the reconstructed data after receiving l packets at the sink for different m using the OMP reconstruction algorithm, with $S_{meas} = 10^3$, $k = 5$, $N = 256$ and $n = 512$ [9, 10].

Figure 6.5 depicts the CDF based on the probability of having at least $RSNR_i >$ 100dB for the number of coded packets r required by the sink for an accurate reconstruction using the OMP algorithm. We observe that for $r < 25$, almost none of the original sensory data could be reconstructed for any of the measurement sizes m chosen. However, as the number of received packets increases, the reconstruction probability improves almost similarly for $m \in \{64, 128, 256\}$. Complete reconstruction is, therefore guaranteed, at $r \approx 90$ with a high RSNR. Unfortunately, with smaller measurement sizes (e.g., $m = 32$), the probability converges to approximately 0.8 regardless of the number of packets received. Based on these simulation results, we fix the number of measurements after the temporal compression to $m = 64$ for the remaining of our simulations campaign, as it can ensure overall improved system performance and optimized decoding complexity.

"Figure 6.6 illustrates the CDF of different cluster sizes N (number of sensor nodes),

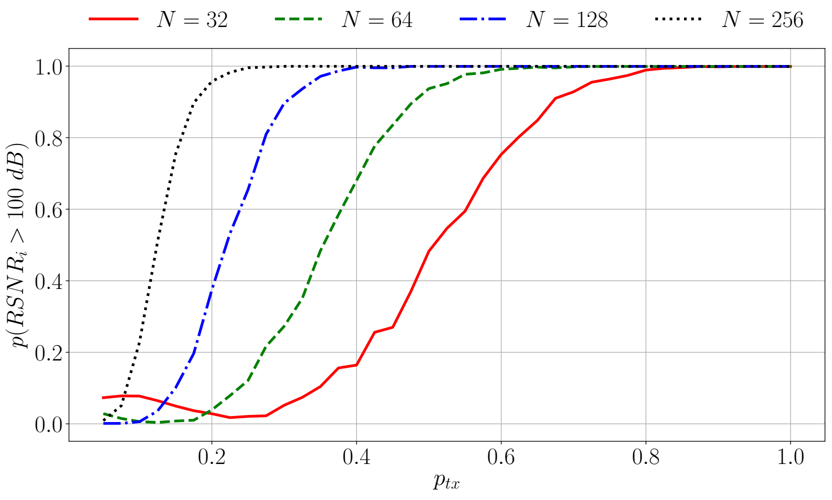


Figure 6.6: Decoding probability as a function of the transmission probability p_{tx} with $N \in \{32, 64, 128, 256\}$, $S_{\text{meas}} = 100/\Delta p_{tx}$ [9,10].

when $\text{RSNR}_i > 100\text{dB}$ as a function of the transmission probability p_{tx} of the sensor nodes. We show that for all values of N, the decoding probability reaches one after a certain transmission probability p_{tx}. Consequently, it is possible to recover the data from all the nodes even if a subsection of them is transmitting. We noticed also an inverse proportional relationship between N and the transmission probability, meaning, one needs a higher N and a lower probability to have $P(\text{RSNR}_i > \text{RSNR}_{min}) = 1$. This is due to the fact that the number of data vectors l needed at the sink for the spatial reconstruction scales logarithmically with N for the OMP algorithm. Therefore, the transmission probability scales with $p_{tx} \sim \log(N)/N$. Another factor is the underlying binomial distribution of r, where the probability to receive more than l data vectors depends on N as well" [10].

As for determining the appropriate network coding coefficient distributions that satisfy the RIP with a sufficiently high probability, we propose the evaluation of the one-cluster topology using coefficients drawn from the common compressed sensing distributions, namely the Bernoulli and Gaussian distributions, along with the basic Random subselection case, where the cluster head selects $l = r$ packets at random to relay to the sink, instead of recombining them with the data coming from the other sensors. Figure 6.7 depicts the CDF of the number of coded packets r needed at the sink to reconstruct the original data that was combined with intra-cluster network coding coefficients from the aforementioned distributions. As the theory of compressed sensing stipulates, the performance of both Bernoulli and Gaussian matrices are similar and

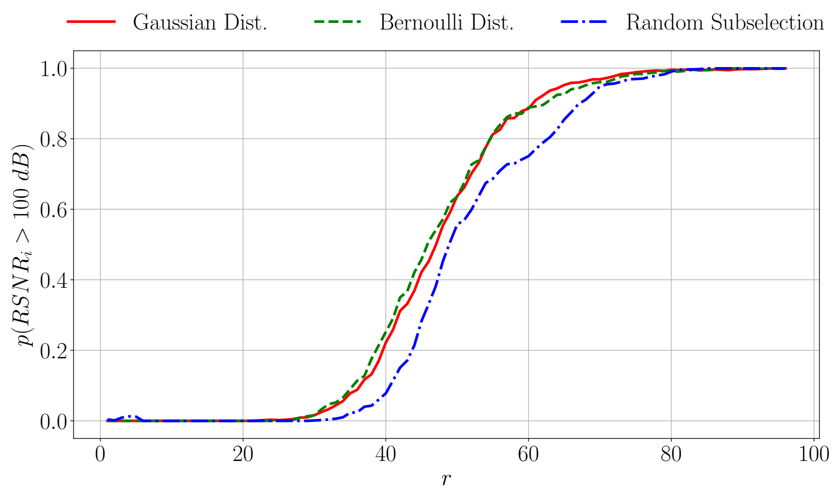


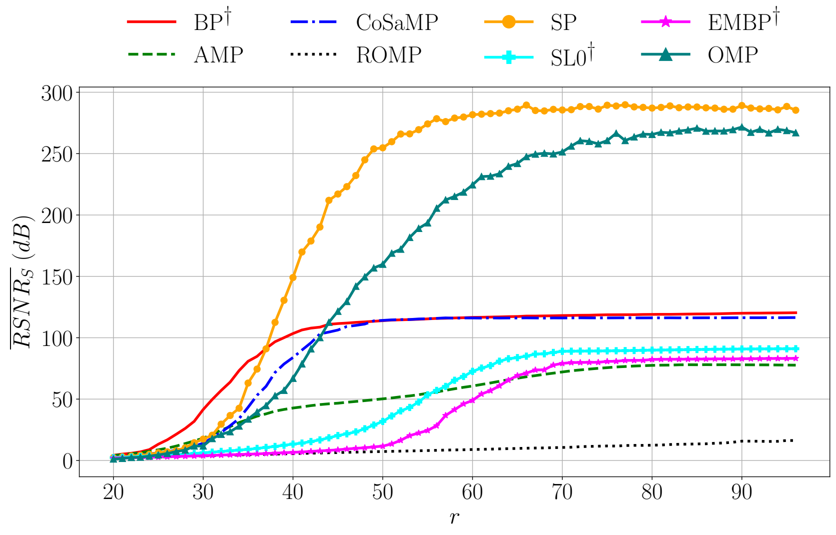
Figure 6.7: Decoding probability over r, the number of packets received at the sink for different intra-cluster network coding coefficients or random subselection at the cluster head with the *OMP* algorithm, where $S_{meas} = 10^3$, $k = 5$, $N = 256$, $n = 512$ and $m = 64$ [9,10].

allow a reconstruction starting from $r = 26$. Nonetheless, we observe with random subselection that data reconstruction starts from $r = 36$ for the same reconstruction probability. This gap persists until $r = 80$, where the decoding probabilities converge to one for all the coding matrix distributions. This result is practically relevant since the Bernoulli coefficients have a lower storage cost than the Gaussian ones. Additionally, recombining various source node data provides a better decoding performance for lower r values than merely selecting data vectors at random. Possible reasons for this include, that Bernoulli or Gaussian distributed sensing matrices are better conditioned than the random selection operator, as well as that the sum of multiple data vectors contains more information than a single one.

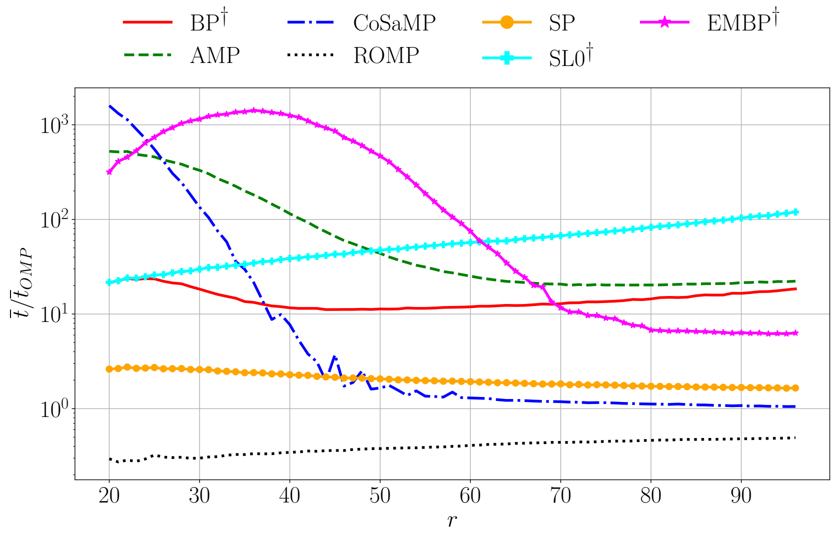
6.4.3 Performance of Reconstruction Algorithms

In order to choose the best-suited algorithm for our data model and topology, we evaluate the KL1p algorithms in terms of their reconstruction SNR and the computation time for the single-cluster topology.

Figure 6.8a compares the average RSNR, denoted by $\overline{\mathrm{RSNR}}_S$, for the benchmark reconstruction algorithms after the spatial reconstruction as a function of the number of packets r received at the sink. We observe that the SP and OMP algorithms achieve outstanding results compared to the other algorithms, with a respective average of



(a) Average RSNR$_S$



(b) Average time $\bar{t}$ needed for spatial recovery with respect with the OMP average time, $\bar{t}_{OMP}$

Figure 6.8: Comparison of the reconstruction algorithms for $S_{\text{meas}} = 1000/S_{\text{meas}} = 100(\dagger)$, $k = 5$, $N = 256$, $n = 512$, $m = 64$, $l = 96$ and $\mu = 1.4$ [9,10].

RSNR$_S$ = 280 dB and RSNR$_S$ = 260 dB with $r = 90$. The remaining benchmark algorithms' performance does not exceed an average RSNR$_S$ of 120.

Additionally, it is crucial to determine the fastest algorithm in completing the reconstruction in order to draw our conclusions on the best available options, and their trade-offs (if there are any). In order to be able to plot the results altogether, we needed to scale the BP and the EMBP algorithms by showing the results for $S_{meas} = 10^2$ not

$S_{meas} = 10^3$ as it is the case for the other algorithms. Moreover, the OMP algorithm completion time was by far the shortest, which has led us to think of comparing the benchmark algorithms to the OMP completion time, denoted by $\bar{t}_{OMP}$.

Figure 6.8b shows the average computation time needed for reconstruction compared to the OMP algorithm's computation time. We observe that the SP algorithm, which has a similar performance to the OMP based on Figure 6.8a, takes around twice as long compared to OMP. Therefore, we conclude that the OMP is best suited in single-cluster scenarios, as it cuts down the simulation time, while slightly compromising the average SNR performance compared to the SP algorithm.

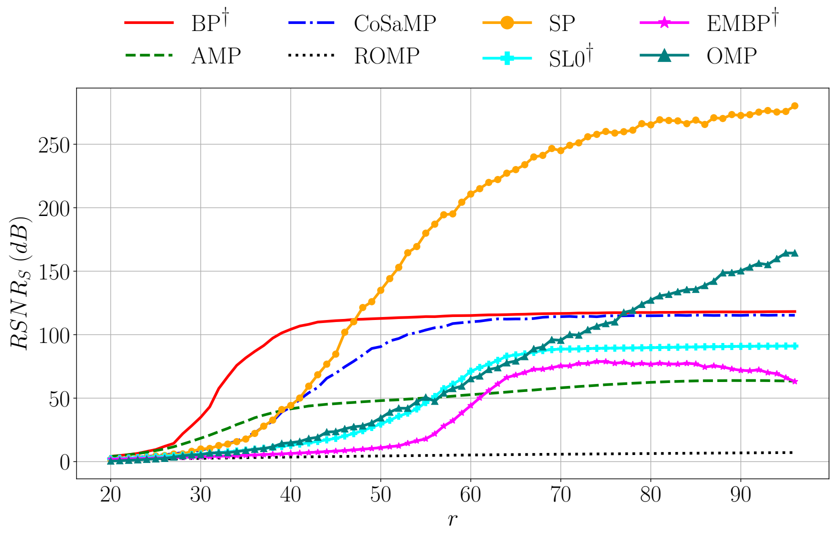


Figure 6.9: Average RSNR_S after spatial recovery over number of packets received at the sink with inter cluster recombinations, different algorithms, $S_{\text{meas}} = 1000/S_{\text{meas}} = 100(\dagger)$, $k = 5$, $N = 256$, $n = 512$, $m = 64$, $l = 96$ and $\mu = 1.4$ [9, 10].

"On the other hand, to simulate the inter-cluster network coding, we let the CH form l recombinations of the spatially compressed inner cluster data. As shown in Figure 6.9, this massively influences the performance of most algorithms. Usually, more packets are needed at the sink to achieve the same RSNR_S as without recombinations. The reason for the decreased performance may be attributed to perturbations in the data vectors and in the sensing matrix. These initially occur during recombinations at CH_1, as well as the additional matrix multiplication at the decoder. Those errors arise in consequence of rounding during the calculations, since we represent coefficients of an infinite field with a fixed size `double` standard. We also note that only the BP algorithm seems to be unaffected and it performs best with r. However, since its complexity scales with N^3 and n^3, it appears not to be practical for a real-life scenario. Since the SP algorithm shows the best performance considering the absolute RSNR$_S$, we rely on it when analyzing scenarios where recombinations are taking place" [10].

6.4.4 Performance Evaluation of the Four-Cluster Topology

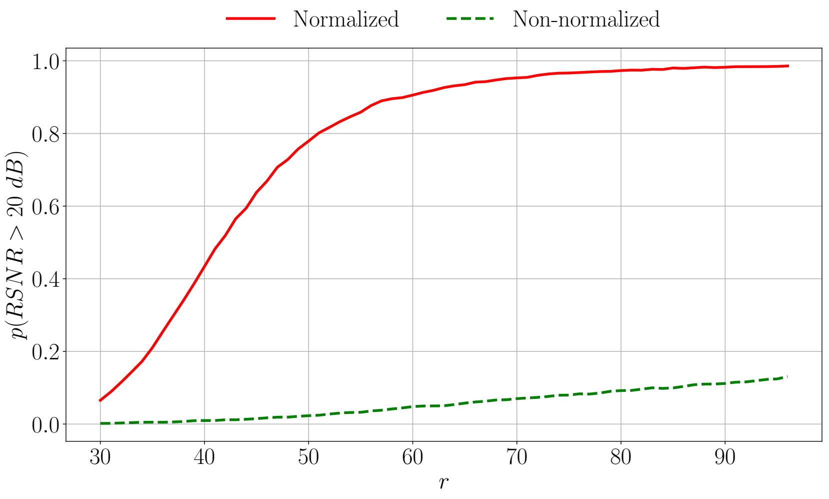


Figure 6.10: Decoding probability over number of packets r received at the sink with (non-)normalized coefficients, two recombinations, SP, $S_{meas} = 1000$, $k = 5$, $n = 512$, $m = 64$, $\mu_j = 1.4\ \forall j = 1, 2, \cdots$ and RSNR = 150 dB [9, 10].

We propose to determine the correct random coefficients to allow efficient recombinations using real network coding. Namely, we investigate the impact of normalized Gaussian coefficients on the overall reconstruction SNR of the four-cluster topology. Furthermore, we determine the JoComCo scheme gains based on the sparsity level of the original sensory data considered compared to a set of benchmark approaches.

It is crucial to determine the correct normalization of the network coding coefficients when applying several recombination (recoding) steps on top of the compressed sensing operations previously performed on the original sensory data. These coefficients are either drawn properly normalized or simply chosen from $\mathcal{N}(0, 1)$ without normalization.

Figure 6.10 illustrates the impact of Gaussian coefficient normalization on the RSNR of the overall topology. We observe that non-normalized coefficients perform extremely badly, with a probability of achieving a reconstruction SNR larger than 20dB not exceeding 0.13 when considering a large number of recombined packets $r = 96$. Figure 6.11(a) confirms this result and shows that the new coefficient distribution cannot be fitted properly in a Gaussian distribution. However, when the coefficient is properly normalized, it is guaranteed that $r \approx 90$ is sufficient to finish decoding using the SP algorithm

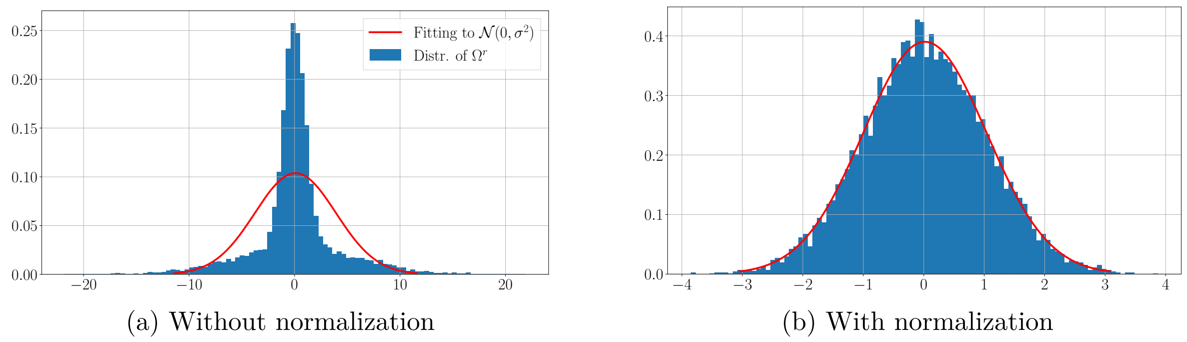


Figure 6.11: Distribution of network coding matrix elements of a single measurement sequence with two recombinations fitted to a Gaussian distribution, with and without normalization, with $r = 96$, $k = 5$, $n = 512$, $m = 64$ and $RSNR = 150dB$, and the SP reconstruction algorithm used [9, 10].

with a $RSNR > 20$ dB, with a very high probability. This can be well described by Figure 6.11(b), where the distribution of the normalized coefficients is fitted in the Gaussian distribution function.

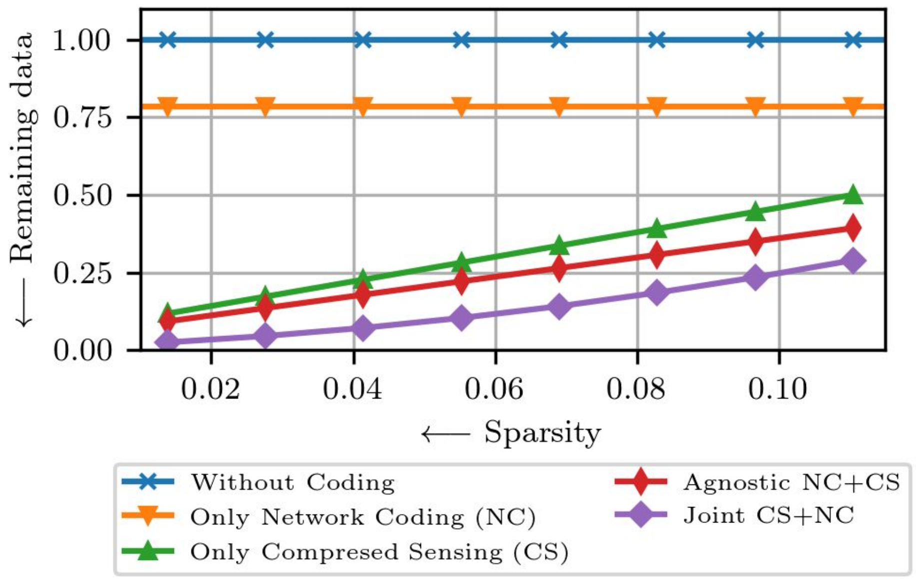


Figure 6.12: Received normalized data by the sink for the four-cluster topology setting based on the original readings sparsity factor, when $N_j = 128$, $\mu_j = 1.2$, $\forall j \in \mathcal{C}$, $n = 512$, and SNR= 150dB [8].

Figure 6.12 illustrates the normalized data reduction gain as a function of the normalized sparsity of the data considered, for the set of different schemes in the four-cluster topology of Figure 6.2. This data is approximately reconstructed at the receiver with a sufficiently high SNR around 150 dB. We observe that the different levels of sparsity do not impact the number of coded packets transmitted when using network coding. This proves our assumption that network coding is an agnostic coding technique, in which performance does not depend on the data type. Nevertheless, discarding a fraction of around 0.2 from the total transmission thanks to its FEC nature and its ability to discard redundant packets are considered to be beneficial especially in scenarios where compressed sensing cannot be performed by very low-battery sensors. Furthermore, the scheme where only compressed sensing is applied shows a great data reduction that is linearly dependent on the sparsity. On average, around half of the data in this four-cluster network could be removed for a normalized sparsity of 9.1 thanks to the efficient exploitation of the temporal and spatial correlations by compressed sensing. As for the scenario where compressed sensing and network coding are combined, a slight improvement could be noticed when the data are very sparse, but the gap increases if the data are less sparse, leaving around 0.4 of the data only. Finally, our proposed scheme JoComCo, referred to as "Joint CS+NC", outperforms the agnostic approach, keeping around 0.3 of the data with an original normalized sparsity of 0.1, and around 0.1 when the normalized sparsity is around 0.05. Our scheme design not only allows for reduced data communication but also faster reconstruction thanks to the one-step decoding we previously explained.

6.5 Conclusions

In this chapter, we proposed an in-network computing approach by designing combined compressed sensing and network coding scheme. We unveiled the most important settings for an advantageous combination compared to other state-of-the-art agnostic approaches. We mainly focused on choosing the most efficient coding and sensing matrices, as well as the fastest reconstruction algorithms that guarantee higher reconstruction SNRs. For instance, the network coding matrix should obey all the compressed sensing properties, e.g., the RIP or NSP, despite the fact that it is unaware of the characteristics of the data to be coded. Nevertheless, it is crucial to meticulously choose the network coding matrix coefficients. For example, normalized Gaussian coefficients resolve the problem of the packet header explosion with a high probability. Our implementation results using a set of reconstruction algorithms prove that the SP guarantees the best performance, especially for a large networks. Whereas, it is more convenient to employ OMP for simple topologies such as our single-cluster topology since it is the fastest algorithm and its performance is closer to the one by SP.

JoComCo allows the efficient in-network exploitation the temporal and spatial correlations (also seen as inter and intra correlations) in a WSN. To the best of our knowledge, this is the only scheme that explicitly provides an overall improved gain compared to the agnostic state-of-the-art approaches. We showed an over 90 % decrease in the payload and the number of packets specifically for data that follow the Joint Sparsity Model (JSM-2), such as temperature and humidity readings, while maintaining a low computational complexity thanks to the joint decoding mechanism, and a very high reconstruction SNR [8, 10].

Nevertheless, our scheme cannot cope with extremely repetitive recoding in larger networks with a large number of clusters due to the inevitable multiplication precision loss in the real field. Therefore, it is interesting to improve the scalability of the recoding feature of this approach or design an efficient joint scheme in the finite field.

Chapter 7 Conclusions and Future Directions

The fifth generation of mobile communication systems is rapidly expanding the tasks of both wired and wireless networks. These novel solutions are expected to play crucial roles for both private and corporate uses. This new era, which will connect everyone and everything, is more engaged in the data deluge continuum, since, presently, there are around 50 billion sensors deployed throughout all IoT equipment. Such devices continuously transmit their readings, be it Industry 4.0-enabled robots, self-driving cars, drones, etc. Additionally, we are witnessing a huge volume of video broadcasts and exchanges. We can just look at how video conferencing has evolved due to the COVID-19 pandemic, which dramatically increased its proportional claim on the overall bandwidth in order to carry on with connecting both people and businesses [221]. Keeping up with the management of this new wave of big data should start from the very communications systems that are handling the brunt of these massive transmissions. Studying and understanding the various features of this overwhelming new wave of data has led us to realize that the data packets are actually quite heterogeneous. When looking at such messages, even just the packet size distributions are seemingly quite random. On the other hand, the information carried within such messages could exhibit significant levels of correlations, especially in the omnipresent wireless sensor networks.

Network coding advocates for a new concept of the communication paradigm that heavily relies on a point-to-point type of linear coding. Instead, network coding allows the intermediate nodes to perform recoding without resorting to decoding, which resulted in tremendous improvements in the network robustness, resilience, and throughput, to name a few. On the other hand, the compressed sensing technique exploits the correlations of signals to allow for the exact or approximate recovery of a high-dimensional sparse signal from low-dimensional measurements, thus breaking with the Shannon-Nyquist theory and the standard rules of linear algebra. From a communication point of view, compressed sensing could serve as a means to reducing data transmissions and storage. Both techniques are of significant importance for the new generation of communications. Plugging them in the appropriate 5G use-cases guarantees substantial network performance gains. Our contributions in this dissertation are twofold: the first significantly reduces the padding overhead resulting from performing RLNC on real-life data, known to have unequal sizes, and the second exploits the correlations of the sensory data to reduce the payload and the number of transmissions.

The first contribution consists of exploiting the heterogeneity of the data to transmit, i.e., the randomness in packet size distributions, by proposing novel mechanisms that reduce the resulting padding overhead when employing FEC codes, namely RLNC, since it is the unique code that enables recoding at intermediate nodes. Therefore, we proposed a set of schemes that operate on macro-symbols, which represent subsets of the packets instead of the entire packets. The following approaches represent significant improvements:

- progressive shortening
- deterministic shifting
- Random Shift and XOR (RaSOR)

Specifically, progressive shortening is a unique RLNC-based code that generates unequal-sized coded packets with sizes almost matching that of the original packets', since it operates on individual macro-symbols that involve minimal padding overhead [4]. Interestingly, it is also capable of healing a generation from a padding overhead resulting from the redundancy in the transmissions during recoding. Consequently, it creates unequal-length recoded packets as well, which linearly increases with the number of hops using RLNC [6]. However, despite our efforts in reducing its expensive decoding complexity with sub-decoders, which consist of smaller decoding matrices compared to the ones used in MS RLNC, its throughput remains low compared to conventional RLNC. Furthermore, a robust policy is needed to raise awareness about when to reduce the sizes of the following coded packets, instead of resorting to sending full-length ones, especially when lacking feedback.

The macro-symbol shifting idea provides more flexibility when combining the individual macro-symbols, thus favoring the creation of less coded packets compared to the original generation size [4, 5]. This idea is inspired by the chain and fragmentation scheme [169], but we proposed instead the employment of macro-symbols. The lower bound of the required coded packets under this setting is defined by $\Delta_{\max}^{det}$ as $\left\lceil \sum_{i=1}^{N} \Lambda_i \Big/ \Lambda_{\max} \right\rceil$, which could be significantly lower if the potential RLNC padding overhead is high. Moreover, the idea of actively shifting the packets before creating a new coded packet has added an extra computational complexity for decoding, since the pattern of the coefficients in the decoding matrix is constantly updated. Moreover, the average performance was similar to the random shifting approach introduced in Chapter 5. This has led us to discard the idea of drawing random coefficients from a finite field, instead, we perform XORing operations on the shifted macro-symbols in order to reduce the overall complexity.

The related approach is called Random Shift and XOR (RaSOR), which is based on shifting the unequal-sized packets then XORing before the creation of every coded packet [30]. RaSOR also has a linear encoding/decoding complexity, which makes it best suited for power-constrained devices, like sensors. Moreover, this scheme comes with some varieties that exclusively depend on the coding generation in question. As far as we know, the only drawback of the RaSOR scheme is its lack of recoding support, since the coded packets have equal sizes that suppress the creation of sufficiently innovative packets with shifted-XORing. Random shifting achieves, on average, the same performance as the deterministic scheme. Despite its relatively low performance, it constitutes a free and easy to apply code to protect the data transmissions, as well as its storage [222].

Although my Ph.D. journey must come to an end, this does not mean that all my (our) research ideas have been fully exploited and published. On the contrary, we have consistent material related to the topics of the dissertation that has not yet been submitted for review. For instance, we are actively working on the first RLNC technique survey that could serve as a guide for students in communication networks, as well as researchers from academia and industry. Parts of this work were introduced in Chapter 2. Moreover, we investigated the performance of the progressive shortening scheme under different coefficient sparsity levels and compared it to conventional RLNC. Furthermore, we proposed an overlapping-generation mechanism to overcome the issue of the macro-symbols that are quickly discarded from the encoding process of the progressive shortening. Since the progressive shortening approach has a higher decoding computational complexity, we decided to target the optimization of decoding via higher parallelization.

There are also various interesting directions for future works that could employ an online sliding window to improve the encoding and decoding complexities at a granularity of individual packets, leading to reduced network latencies [223, 224]. Additionally, for further gains, it is important to examine the padding overhead reduction by proposing intra-coding of the macro-symbols inside the space that is usually occupied by zero-padding in conventional RLNC. As an example, enhancing the robustness of the data by incorporating coded versions of the macro-symbols within a specific original packet before performing RLNC. Furthermore, we would like to include our macro-symbol-based code in the Kodo library of Steinwurf, and give free access to students to manipulate the code, in order for them to be able to build upon it for their personal projects.

Our second contribution was mainly to the design and implementation of an efficient joint compressed sensing and real network coding approach called JoComCo. JoComCo surpasses the overall performance of the state-of-the-art approaches, especially in terms of compression gain. Investigating the characteristics needed for a fruitful design enabled us to exploit the recoding feature of network coding in the real field to further extend its adoption in slightly larger networks. However, it remains a challenging task to provide a scalable design for large-scale recoding, as the operations on floating points in the real field may lead to a deviation from the original values, thus blocking potential gains.

This is the reason why we were interested in 1-bit compressed sensing, which inherits the compressed sensing model but only retains the signs of the measurements [225]. We believe that such a variant could enable a smoother joint design with RLNC, instead of the approaches discussed in Chapter 6. Our results show a high SNR for Gaussian and Rademacher matrices, as well as a faster convergence time for the Binary Iterative Hard Thresholding (BIHT) [226] algorithm when Rademacher matrices are used. As its name suggests, 1-bit compressed sensing reduces the data sets into mere few bits, which could be an interesting approach for applications where quick decisions have to be made before the data becomes outdated and discarded. Moreover, investigating opportunities for deep learning [227] to alleviate core compressed sensing challenges, including the search for adequate and more effective sparsifying basis, as well as measurement matrices, will definitely lead to significantly higher compression gains. This is expected to favor the low-latency and low-complexity reconstruction process while maintaining the accuracy of the data. Such a combination has the potential of enabling distributed partial decoding as a further step. On the other hand, it is of tremendous interest to investigate the possibility of a joint design of RaSOR with compressed sensing. By relying on tunable-size sensing matrices, which could bring a broader interest in applying it in heterogeneous IoT-based networks (such as road monitoring), it could have a high potential application interest for unequal but correlated data transmissions.

Eventually, it is crucial to continue the research of post-Shannon communications in order to obtain more efficient transmissions and data management. The ultimate goal would be to enable the recreation of data on-demand, given that we hold the key to functional compression. Could this be a reasonable goal compared to holographic teleportation set as a goal for the advent of 6G [228, 229]? We will hopefully see how such avenues of research turn out in the future...

Appendix A
Proofs for Section 6.4

A.1 Proof of Proposition 1

Proof. We use a simple iteration in order to calculate the rank evolution of the C different combinations/equations before we obtain the maximum rank for a generation with equal-size packets, i.e. $\mathcal{G} = \{P_1, \cdots, P_N\}$, where $P_i = \{\mathbf{s_{ik}}, k \in \{1, \cdots, L_{\max}\}\}$. To do so, we choose and fix the $(N-1)$ elements from different packet sets, for simplicity $\mathbf{s_{j1}}$, $j \in \mathsf{F}_i = \{1, \cdots, N\} \backslash \{i\}$, and we vary the remaining set P_i within each iteration to count the rank evolution accordingly. The resulting value of the linear equation obtained by XORing these elements is $\mathbf{V_{ik}}$:

$$\mathbf{V_{ik}} = \bigoplus_{j \in \mathsf{F}_i} \mathbf{s_{j1}} \oplus \mathbf{s_{ik}}, \forall k \in \{1, \cdots, L_{\max}\}.$$

Therefore, $rk\{V_{11}, \cdots, V_{1L_{\max}}\} = L_{\max}$ for $i = 1$. It is trivial that the rank of a linear system of equations obtained using XOR operations increases if we add a new equation that can be written as $z + y$ where y represents unknowns that have not appeared in other early equations. As a result, the rank increases by 1 iff a new unknown is brought with a new equation. To continue the counting process for $i \in \{2, \cdots, N\}$, we iterate through $k \in \{2, \cdots, L_{\max}\}$ since $\mathbf{V_{i1}} = \bigoplus_{j \in \mathsf{F}_i} \mathbf{s_{j1}} \oplus \mathbf{s_{i1}} = \mathbf{V_{11}}$. Consequently,

$$rk(\mathcal{C}) = rk\Big\{ \underbrace{\mathbf{V_{11}}, \cdots, \mathbf{V_{1L_{\max}}}}_{L_{\max}}, \underbrace{\cancel{\mathbf{V_{21}}}, \cdots, \mathbf{V_{2L_{\max}}}}_{(L_{\max}-1)}, \cdots, \underbrace{\cancel{\mathbf{V_{N1}}}, \cdots, \mathbf{V_{NL_{\max}}}}_{(L_{\max}-1)} \Big\} = L_{\max} + (L_{\max} - 1) \cdot (N - 1)$$

Hence $rk(\mathcal{C}) = NL_{\max} - N + 1$. This completes the proof of Proposition 1. □

A.2 Proof of Proposition 2

Proof. This follows the same reasoning except that the packets do not have the same sizes. Let $P_i = \{\mathbf{s_{ij}}, j \in \{1, \cdots, L_i\}\}$ and $|\mathcal{K}| = n$. However, in this case, a non-full size packet has the possibility of not appearing in these iterations. As such,

$$\mathbf{V_{ik}} = \bigoplus_{j \in \mathsf{F}_i} \mathbf{s_{j1}} \oplus \mathbf{s_{ik}}, \forall k \in \{1, \cdots, L_i\}.$$

For ease of notations let the first n packets be full size, then:

$$rk(\mathcal{C}) = rk\Big\{ \underbrace{\mathbf{V_{11}}, ..., \mathbf{V_{1L_{\max}}}}_{L_{\max}}, \underbrace{\cancel{\mathbf{V_{21}}}, ..., \mathbf{V_{2L_{\max}}}}_{(L_{\max}-1)}, ...,$$

$$\underbrace{\cancel{\mathbf{V_{n1}}}, ..., \mathbf{V_{nL_{\max}}}}_{L_{\max}-1}, \underbrace{\mathbf{V_{(n+1)1}}, ..., \mathbf{V_{(n+1)L_{(n+1)}}}}_{L_{(n+1)}} ..., \underbrace{\mathbf{V_{N1}}, ..., \mathbf{V_{NL_N}}}_{L_N} \Big\}$$

$$= L_{\max} + \sum_{l=2}^{n} (L_{\max} - 1) + \sum_{t=n+1}^{N} L_l = \sum_{l=1}^{N} L_l - (n-1),$$

which concludes the proof of Proposition 2. □

A.3 Selecting the Modifier μ_j via a Fixed Deviation

The transmission probability of the common source nodes in cluster j can be denoted as

$$p_{tx} = \mu_j (l_j - 1)/(N_j - 1),$$

and is equally to the reception probability at the CH in case of no link errors, so $p_{rx} = p_{tx}$. The probability to receive at least l_j data vectors at the CH is given by a binomial distribution:

$$p\left(r \geq l_j - 1\right) = 1 - \sum_{i=0}^{l_j-2} \binom{N_j - 1}{i} p_{rx}^{i} (1 - p_{rx})^{N_j - 1 - i}.$$

We want to fix the transmission by changing μ_j, so that the CH does not receive less than $l_j - 1$ data vectors with a fixed deviation from the mean of this binomial distribution (with $l_j - 1 = a, N_j - 1 = b$):

$$a \overset{!}{=} \mu_j a - \kappa\sigma \quad \rightarrow \quad \mu_j - 1 \overset{!}{=} \frac{\kappa\sigma}{a}, \quad \kappa \in \mathbb{R}.$$

The deviation of the binomial distribution is given by:

$$\sigma^2 = \mu_j a \left(1 - \mu_j \frac{a}{b}\right).$$

Then by joining both equations via the deviation σ it follows that:

$$0 = \mu_j^2 \left(1 + \frac{\kappa^2}{b}\right) - \mu_j \left(2 + \frac{\kappa^2}{a}\right) + 1.$$

This leads to a the solution of $\mu_j > 1$:

$$\mu_j = \frac{\left(2 + \frac{\kappa^2}{a}\right) + \sqrt{\left(2 + \frac{\kappa^2}{a}\right)^2 - 4\left(1 + \frac{\kappa^2}{b}\right)}}{2\left(1 + \frac{\kappa^2}{b}\right)}.$$

The desired deviation can be, for example, chosen by approximating the binomial with a Gaussian distribution. For $\kappa = 2$ then it is given with the Gaussian confidence intervals:

$$p\left(r \geq l_j - 1\right) \approx 1 - 0.022 = 0.978.$$

Appendix B Network Emulation with Compressed Sensing

The Communication Networks Emulator (ComNetsEmu) was built for the sake of a collaborative research book that was prepared by most of the researchers at the Deutsche Telekom Chair of Communication Networks [1]. It encloses our latest results in a vast range of research fields. We implemented a simple compressed sensing framework for the ComNetsEmu emulator [11]. In order to remain consistent and harmonized with the book's content, we considered each sensor node to be used as an independent Docker container that has powerful computational capabilities to perform compressed sensing independently. In the following, we present the example of a single-cluster topology, where the sensory data can undergo either a DCT transform for obtaining a sparse representation, or an over-complete trained dictionary. We note that the content of the appendix has appeared entirely in [11].

B.1 Implementation in ComNets Emulator

We note that the *python-sklearn* and *python-numpy* dependencies are required for running the application in the emulator. The considered example can be found in the *compressed-sensing* directory of the ComNetsEmu. We employ the dataset from 54 sensors provided by Intel lab [230]. The dataset contains environmental data, e.g., temperature, light, and humidity values, which are sampled every 31 seconds. Without loss of generality to other types of sensed data, we propose to use the temperature readings. We also note that this is the most reliable library that we could find for our implementation, despite the missing or truncated values as it is mentioned in the Intel Lab Data webpage.

The scenario we are proposing is depicted in Figure B.1 and consists of six sensors in one cluster where each communicates its compressed readings to the sink that is responsible for the reconstruction procedure.

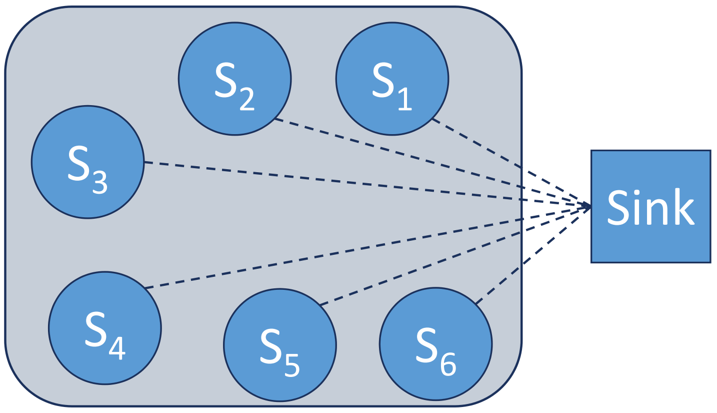


Figure B.1: An example of single cluster scenario where one sensor transmits its compressed readings to a sink [11].

Specifically, every sensor sparsifies and compresses every set of samples $S = 100$, containing each $n = 80$ readings, seen as an $S \times n$ matrix of original readings, into $m = 40$ measurements per sample, i.e., the data is reduced by half. The sink is aware of the sparsification and the compression details, and performs the OMP algorithm to approximate the original sensory data. We propose the evaluation the reconstruction using Mean Squared Error (MSE) scores, which is known to give an efficient approximation of the reconstructed data error.

B.2 Using DCT for Data Sparsification

The command in the listing below is required to be able to run the scenario:

```
$ sudo python3 topo.py 6 --dct 1
```

Listing B.1: Setup of the compressed sensing scenario using overcomplete dictionary learning.

Furthermore, Listing B.2 shows the connections established between the sensors (containers) and the sink [11]:

```
Head: Setup socket on ('10.0.0.21', 8003)
```

```
Head: Now connected to:  ('10.0.0.1', 51770)
Head: Now connected to:  ('10.0.0.2', 38404)
Head: Now connected to:  ('10.0.0.3', 35576)
Head: Now connected to:  ('10.0.0.4', 33658)
Head: Now connected to:  ('10.0.0.5', 60990)
Head: Now connected to:  ('10.0.0.6', 40766)
```

Listing B.2: Detail of the output from Listing B.1

The extended output of Listing B.1 is displayed in Listing B.3, where the MSE of the reconstructed data received from all six sensors is displayed.

```
Head: Shape received data buffer:  (6, 20, 1)
Head: Decompression with DCT

Head: Shape reconstructed data buffer:  (6, 100, 1)
Head: MSE for Sensor1 : 2.4971498629114364
Head: MSE for Sensor2 : 3.205683707754018
Head: MSE for Sensor3 : 1.4509731095228484
Head: MSE for Sensor4 : 2.5766685412602643
Head: MSE for Sensor5 : 3.3931169852261265
Head: MSE for Sensor6 : 2.497400591976876

*** Head has finished
*** Node1 logs
Node1: Connected to:  ('10.0.0.21', 8003)
Node1: Compression with dct
Node1: Shape original data:  (100,)
Node1: Shape compressed data:  (40,)
Node1: closed connection to:  ('10.0.0.21', 8003)

*** Node2 logs
Node2: Connected to:  ('10.0.0.21', 8003)
Node2: Compression with dct
Node2: Shape original data:  (100,)
Node2: Shape compressed data:  (40,)
Node2: closed connection to:  ('10.0.0.21', 8003)
```

```
*** Node3 logs
Node3: Connected to:  ('10.0.0.21', 8003)
Node3: Compression with dct
Node3: Shape original data:  (100,)
Node3: Shape compressed data:  (40,)
Node3: closed connection to:  ('10.0.0.21', 8003)

*** Node4 logs
Node4: Connected to:  ('10.0.0.21', 8003)
Node4: Compression with dct
Node4: Shape original data:  (100,)
Node4: Shape compressed data:  (40,)
Node4: closed connection to:  ('10.0.0.21', 8003)

*** Node5 logs
Node5: Connected to:  ('10.0.0.21', 8003)
Node5: Compression with dct
Node5: Shape original data:  (100,)
Node5: Shape compressed data:  (40,)
Node5: closed connection to:  ('10.0.0.21', 8003)

*** Node6 logs
Node6: Connected to:  ('10.0.0.21', 8003)
Node6: Compression with dct
Node6: Shape original data:  (100,)
Node6: Shape compressed data:  (40,)
Node6: closed connection to:  ('10.0.0.21', 8003)
```

Listing B.3: Detail of the output from Listing B.1

We observe that despite the close geographical placement of the sensors, and the similar temperature readings during the sensing times, the reconstruction gives a different value of the MSE per sensor, which is due to the several potential errors within the data itself.

B.3 Using a Trained Dictionary for Data Sparsification

In this scenario, we rely on the training of an over-complete dictionary to use instead of the pre-defined DCT sparsification. The dictionary is obtained using the K-SVD algorithm and a larger set of sensor readings compared to the sets used for compressed sensing. We developed our K-SVD algorithm using the OMP algorithm of the Scikit-learn Application Programming Interface (API). To run this example, the following command needs to be executed:

```
$ sudo python3 topo.py 6
```

Listing B.4: Setup of the compressed sensing scenario using overcomplete dictionary learning.

As the connections establishment between the containers and the sink is similar to the previous DCT scenario, Listing B.4 displays the MSE of the reconstructed data for each of the six sensors [11]:

```
Head: Shape received data buffer:  (6, 100, 40)
Head: Decompression with dictionary

Head: Shape reconstructed data buffer:  (6, 100, 80)

Head: MSE for Sensor1 : 0.18296451461710317
Head: MSE for Sensor2 : 0.5187587073441643
Head: MSE for Sensor3 : 0.17176943538612072
Head: MSE for Sensor4 : 0.18735776249882208
Head: MSE for Sensor5 : 0.3512422779473302
Head: MSE for Sensor6 : 0.16979520531023373

*** Head has finished
*** Node1 logs
Node1: Connected to:  ('10.0.0.21', 8003)
Node1: Compression with dictionary
Node1: Shape original data:  (100, 80)
Node1: Shape sparse data:  (200, 100)
Node1: Shape compressed data:  (100, 40)
```

```
Node1: MSE:  0.1829645146171032
Node1: closed connection to:  ('10.0.0.21', 8003)

*** Node2 logs
Node2: Connected to:  ('10.0.0.21', 8003)
Node2: Compression with dictionary
Node2: Shape original data:  (100, 80)
Node2: Shape sparse data:  (200, 100)
Node2: Shape compressed data:  (100, 40)
Node2: MSE:  0.5187587073441643
Node2: closed connection to:  ('10.0.0.21', 8003)

*** Node3 logs
Node3: Connected to:  ('10.0.0.21', 8003)
Node3: Compression with dictionary
Node3: Shape original data:  (100, 80)
Node3: Shape sparse data:  (200, 100)
Node3: Shape compressed data:  (100, 40)
Node3: MSE:  0.17176943538612072
Node3: closed connection to:  ('10.0.0.21', 8003)

*** Node4 logs
Node4: Connected to:  ('10.0.0.21', 8003)
Node4: Compression with dictionary
Node4: Shape original data:  (100, 80)
Node4: Shape sparse data:  (200, 100)
Node4: Shape compressed data:  (100, 40)
Node4: MSE:  0.18735776249882208
Node4: closed connection to:  ('10.0.0.21', 8003)

*** Node5 logs
Node5: Connected to:  ('10.0.0.21', 8003)
Node5: Compression with dictionary
Node5: Shape original data:  (100, 80)
Node5: Shape sparse data:  (200, 100)
Node5: Shape compressed data:  (100, 40)
Node5: MSE:  0.3512422779473302
```

```
Node5: closed connection to:  ('10.0.0.21', 8003)

*** Node6 logs
Node6: Connected to:  ('10.0.0.21', 8003)
Node6: Compression with dictionary
Node6: Shape original data:  (100, 80)
Node6: Shape sparse data:  (200, 100)
Node6: Shape compressed data:  (100, 40)
Node6: MSE:  0.1697952053102337
Node6: closed connection to:  ('10.0.0.21', 8003)
```

Listing B.5: Detail of the output from Listing B.4

We observe an average MSE varying from 0.169 for Node1 to 0.518 for Node2, which was noticed in the previous example as well. Nevertheless, using a trained dictionary shows remarkable reconstruction improvements of the original data.

B.4 Overcomplete Dictionary Robustness

In order to investigate the robustness of the previously trained dictionary, we propose to train an over-complete dictionary for the data set of each sensor in the network. Then we apply compressed sensing using the same conditions and assumptions previously adopted.

```
Head: Shape received data buffer:  (6, 100, 40)
Head: Decompression with dictionary

Head: Shape reconstructed data buffer:  (6, 100, 80)

Head: MSE for Sensor1 : 0.18296451461710317
Head: MSE for Sensor2 : 0.3831813907447271
Head: MSE for Sensor3 : 0.17085131674783086
Head: MSE for Sensor4 : 0.18582421022630732
Head: MSE for Sensor5 : 0.2531560266775236
Head: MSE for Sensor6 : 0.16862887303364887
```

Listing B.6: Detail of the output from Listing B.4, with a specific trained dictionary for each sensor

Listing B.6 [11] displays the average MSE for each of the sensors in the single-cluster topology. The obtained values are almost identical to the results previously obtained with one trained dictionary in Listing B.5 for all the sensors except for sensor2, where the MSE dropped by around 0.13, and 0.1 for sensor5. This is explained by the fact that the data set employed is not entirely correct, and we needed to clean it from errors and compensate for missing values. Despite this anomaly, using one single dictionary remains a robust and sufficient option as it reduces the dictionary processing and storage at the devices.

Despite the challenging task of finding an efficient sparsification approach for a certain data type, training an over-complete dictionary seems to be the most efficient and robust approach in the presence of a sufficiently large data set. However, the DCT or other pre-defined transformations could lead to closer benefits if the devices performing the compression do not have good computational capabilities and/or large data sets.

Bibliography

[1] Fitzek, F. H., F. Granelli, and P. Seeling (2020) *Computing in Communication Networks: From Theory to Practice*, Academic Press.

[2] Taghouti, M. (2020) *Compressed Sensing*, vol. 1 of *1*, chap. 10, 1 ed., Elsevier, https://cn.ifn.et.tu-dresden.de/compcombook/.

[3] Taghouti, M., D. E. Lucani, M. V. Pedersen, and A. Bouallègue (2016) "On the Impact of Zero-Padding in Network Coding Efficiency with Internet Traffic and Video Traces," in *IEEE 22nd European Wireless Conference, (EW2016)*, pp. 72–77.

[4] Taghouti, M., D. E. Lucani, J. A. Cabrera, M. Reisslein, M. V. Pedersen, and F. H. Fitzek (2019) "Reduction of padding overhead for RLNC media distribution with variable size packets," *IEEE Transactions on Broadcasting*, **65**(3), pp. 558–576.

[5] Taghouti, M., D. E. Lucani, M. V. Pedersen, and A. Bouallègue (2016) "Random Linear Network Coding for Streams with Unequally Sized Packets: Overhead Reduction without Zero-Padded Schemes," in *23rd Int. Conference on Telecommunications, (ICT)*.

[6] Taghouti, M., M. Tömösközi, M. Howeler, D. E. Lucani, F. H. Fitzek, A. Bouallegue, and P. Ekler (2019) "Implementation of network coding with recoding for unequal-sized and header compressed traffic," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, IEEE, pp. 1–7.

[7] Pedersen, M. V., J. Heide, F. H. Fitzek, and T. Larsen (2009) "Pictureviewer-a mobile application using network coding," in *2009 European Wireless Conference*, IEEE, pp. 151–156.

[8] Taghouti, M., A. K. Chorppath, T. Waurick, and F. H. P. Fitzek (2018) "Practical Compressed Sensing and Network Coding for Intelligent Distributed Communication Networks," 14th IEEE International Wireless Communications and Mobile Computing Conference (IWCMC), Limassol, Cyprus.

[9] ——— (2018) "On the Design of a Joint Compressed Sensing and Network Coding Framework," European Wireless Conference, Catania, Italy.

[10] Taghouti, M., M. Tömösközi, T. Waurick, A. K. Chorppath, and F. H. P. Fitzek (2019) "On the Joint Design of Compressed Sensing and Network Coding for Wireless Communications," *Transactions on Emerging Telecommunications Technologies*.

[11] Taghouti, M. and M. Höweler (2020) *In-network Compressed Sensing*, vol. 1 of *1*, chap. 22, 1 ed., Elsevier.

[12] Seeling, P. and M. Reisslein (2012) "Video Transport Evaluation With H.264 Video Traces," *IEEE Communications Surveys & Tutorials*, **14**(4), pp. 1142–1165.

[13] "Arizona State University's video traces library," .
URL `http://trace.eas.asu.edu/tracemain.html`

[14] KL1p (2012), "KL1p - a portable C++ library for compressed sensing," .
URL `http://kl1p.sourceforge.net/home.html`

[15] Miorandi, D., S. Sicari, F. De Pellegrini, and I. Chlamtac (2012) "Internet of things: Vision, applications and research challenges," *Ad hoc networks*, **10**(7), pp. 1497–1516.

[16] Du, R., P. Santi, M. Xiao, A. V. Vasilakos, and C. Fischione (2018) "The sensable city: A survey on the deployment and management for smart city monitoring," *IEEE Communications Surveys & Tutorials*, **21**(2), pp. 1533–1560.

[17] Mainwaring, A., D. Culler, J. Polastre, R. Szewczyk, and J. Anderson (2002) "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pp. 88–97.

[18] Zhang, Z., Y. Xiao, Z. Ma, M. Xiao, Z. Ding, X. Lei, G. K. Karagiannidis, and P. Fan (2019) "6G wireless networks: Vision, requirements, architecture, and key technologies," *IEEE Vehicular Technology Magazine*, **14**(3), pp. 28–41.

[19] Giordani, M., M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi (2020) "Toward 6g networks: Use cases and technologies," *IEEE Communications Magazine*, **58**(3), pp. 55–61.

[20] (2020), "Cisco Annual Internet Report (2018–2023) White Paper," .
URL `https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html`

[21] (2017), "The Zettabyte Era—Trends and Analysis," .
URL `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html`

[22] Osseiran, A., F. Boccardi, V. Braun, K. Kusume, P. Marsch, M. Maternia, O. Queseth, M. Schellmann, H. Schotten, H. Taoka, et al. (2014) "Scenarios for 5G mobile and wireless communications: the vision of the METIS project," *IEEE communications magazine*, **52**(5), pp. 26–35.

[23] Ahlswede, R., N. Cai, S.-Y. R. Li, , and R. W. Yeung (2000) "Network Information Flow," *IEEE Trans. on Info. Theory*, **46**(4).

[24] Ho, T., R. Koetter, M. Médard, D. R. Karger, and M. Effros (2004) "The Benefits of Coding over Routing in a Randomized Setting," in *Proc. of the IEEE International Symposium on Information Theory (ISIT'04)*.

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

[26] Candès, E. J. et al. (2006) "Compressive sampling," in *Proceedings of the international congress of mathematicians*, vol. 3, Madrid, Spain, pp. 1433–1452.

[27] Candès, E. J., J. Romberg, and T. Tao (2006) "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on information theory*, **52**(2), pp. 489–509.

[28] Candes, E. J. and T. Tao (2005) "Decoding by linear programming," *IEEE transactions on information theory*, **51**(12), pp. 4203–4215.

[29] Aharon, M., M. Elad, and A. Bruckstein (2006) "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on signal processing*, **54**(11), pp. 4311–4322.

[30] Taghouti, M., D. E. Lucani, and F. H. Fitzek (2017) "Random shift and XOR of unequal-sized packets (RaSOR) to shave off transmission overhead," in *Information Sciences and Systems (CISS), 2017 51st Annual Conference on*, IEEE, pp. 1–6.

[31] Taghouti, M., D. Lucani, F. H. Fitzek, and A. Bouallegue (2017) "Random Linear Network Coding Schemes for Reduced Zero-Padding Overhead: Complexity and Overhead Analysis," in *European Wireless 2017; 23th European Wireless Conference; Proceedings of*, VDE, pp. 1–7.

[32] Ludwig, S., M. Karrenbauer, A. Fellan, H. D. Schotten, H. Buhr, S. Seetaraman, N. Niebert, A. Bernardy, V. Seelmann, V. Stich, A. Hoell, C. Stimming, H. Wu, S. Wunderlich, M. Taghouti, F. Fitzek, C. Pallasch, N. Hoffmann, W. Herfs, E. Eberhardt, and T. Schildknecht "A 5G Architecture for the Factory of the Future," pp. 1409–1416.

[33] Wu, H., I. A. Tsokalo, M. Taghouti, H. Salah, and F. H. P. Fitzek (2019) "Compressible Source Separation in Industrial IoT Broadband Communication," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 619–624.

[34] Karrenbauer, M., S. Ludwig, H. Buhr, H. Klessig, A. Bernardy, H. Wu, C. Pallasch, A. Fellan, N. Hoffmann, V. Seelmann, M. Taghouti, et al. (2019) "Future industrial networking: from use cases to wireless technologies to a flexible system architecture," *at-Automatisierungstechnik*, **67**(7), pp. 526–544.

[35] et al., M. K. (2018) "Towards a Flexible Architecture for Industrial Networking," in *23th VDE/ITG Conference on Mobile Communication (23. VDE/ITG Fachtagung Mobilkommunikation)*, VDE.

[36] Klessing, H., S. Ludwig, M. Karrenbauer, H. Schotten, H. Wu, M. Taghouti, F. H. P. Fitzek, and P. T. Lozano (2021) "A Factory of the Future Reference Network Architecture," *submitted to IEEE Communications Magazine*.

[37] Taghouti, M. (2021) "PhD Forum: Padding Overhead Reduction in Random Linear Coded Variable Size Media Streams," in *22nd IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, IEEE.

[38] Ahlswede, R., N. Cai, S.-Y. R. Li, and R. W. Yeung (2000) "Network Information Flow," *IEEE Transactions on Information Theory*, **46**(4).

[39] Bassoli, R., H. Marques, J. Rodriguez, K. W. Shum, and R. Tafazolli (2013) "Network coding theory: A survey," *IEEE Communications Surveys & Tutorials*, **15**(4), pp. 1950–1978.

[40] Douik, A., S. Sorour, T. Y. Al-Naffouri, and M.-S. Alouini (2017) "Instantly decodable network coding: From centralized to device-to-device communications," *IEEE Communications Surveys & Tutorials*, **19**(2), pp. 1201–1224.

[41] Li, S.-Y., R. W. Yeung, and N. Cai (2003) "Linear network coding," *IEEE transactions on information theory*, **49**(2), pp. 371–381.

[42] Koetter, R. and M. Médard (2003) "An algebraic approach to network coding," in *IEEE/ACM Transactions on Networking*, vol. 11, pp. 782 – 795.

[43] Krigslund, J., J. Hansen, M. Hundeboll, D. E. Lucani, and F. H. Fitzek (2013) "CORE: COPE with MORE in wireless meshed networks," in *2013 IEEE 77th vehicular technology conference (VTC Spring)*, IEEE, pp. 1–6.

[44] Hansen, J., J. Krigslund, D. E. Lucani, P. Pahlevani, and F. H. Fitzek (2018) "Bridging inter-flow and intra-flow network coding in wireless mesh networks: From theory to implementation," *Computer Networks*, **145**, pp. 1–12.

[45] Chen, W., K. B. Letaief, and Z. Cao (2007) "Opportunistic network coding for wireless networks," in *Communications, 2007. ICC'07. IEEE International Conference on*, IEEE, pp. 4634–4639.

[46] Kafaie, S., Y. Chen, M. H. Ahmed, and O. A. Dobre (2017) "FlexONC: Joint cooperative forwarding and network coding with precise encoding conditions," *IEEE Transactions on Vehicular Technology*, **66**(8), pp. 7262–7277.

[47] Shen, H., G. Bai, L. Zhao, and Z. Tang (2012) "An adaptive opportunistic network coding mechanism in wireless multimedia sensor networks," *International Journal of Distributed Sensor Networks*, **8**(12), p. 565604.

[48] Traskov, D., M. Médard, P. Sadeghi, and R. Koetter (2009) "Joint scheduling and instantaneously decodable network coding," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, IEEE, pp. 1–6.

[49] Sadeghi, P., D. Traskov, and R. Koetter (2009) "Adaptive network coding for broadcast channels," in *2009 Workshop on Network Coding, Theory, and Applications*, IEEE, pp. 80–85.

[50] Sorour, S. and S. Valaee (2010) "Minimum broadcast decoding delay for generalized instantly decodable network coding," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, IEEE, pp. 1–5.

[51] Li, X., C.-C. Wang, and X. Lin (2011) "On the capacity of immediately-decodable coding schemes for wireless stored-video broadcast with hard deadline constraints," *IEEE Journal on Selected Areas in Communications*, **29**(5), pp. 1094–1105.

[52] ——— (2010) "Throughput and delay analysis on uncoded and coded wireless broadcast with hard deadline constraints," in *2010 Proceedings IEEE INFOCOM*, IEEE, pp. 1–5.

[53] Ho, T., R. Koetter, M. Medard, D. R. Karger, and M. Effros (2003) "The benefits of coding over routing in a randomized setting," .

[54] Dougherty, R., C. Freiling, and K. Zeger (2005) "Insufficiency of linear coding in network information flow," *IEEE transactions on information theory*, **51**(8), pp. 2745–2759.

[55] Li, L., K. Fan, and D. Long (2008) "Nonlinear network coding: a case study," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 2, Citeseer.

[56] Kosut, O., L. Tong, and D. Tse (2009) "Nonlinear network coding is necessary to combat general byzantine attacks," in *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*, IEEE, pp. 593–599.

[57] Chou, P. A., Y. Wu, and K. Jain (2003) "Practical network coding," in *Proceedings of the annual Allerton conference on communication control and computing*, vol. 41, The University; 1998, pp. 40–49.

[58] Heidarzadeh, A. and A. H. Banihashemi (2010) "Overlapped chunked network coding," in *2010 IEEE Information Theory Workshop on Information Theory (ITW 2010, Cairo)*, IEEE, pp. 1–5.

[59] Silva, D. and F. R. Kschischang (2009) "On metrics for error correction in network coding," *IEEE Transactions on Information Theory*, **55**(12), pp. 5479–5490.

[60] Yang, S. and R. W. Yeung (2014) "Batched sparse codes," *IEEE Transactions on Information Theory*, **60**(9), pp. 5322–5346.

[61] Pedersen, M. V., J. Heide, and F. Fitzek (2011) "Kodo: An Open and Research Oriented Network Coding Library," *Lecture Notes in Computer Science*, **6827**, pp. 145–152.

[62] Bioglio, V., M. Grangetto, R. Gaeta, and M. Sereno (2009) "On the fly Gaussian elimination for LT codes," *Communications Letters, IEEE*, **13**(12), pp. 953–955.

[63] Drinea, E., C. Fragouli, and L. Keller (2009) "Delay with network coding and feedback," in *2009 IEEE International Symposium on Information Theory*, IEEE, pp. 844–848.

[64] Steinwurf, "Fifi git repository on github," .
URL `https://github.com/steinwurf/fifi-python`

[65] Heide, J. and D. Lucani (2015) "Composite extension finite fields for low overhead network coding: Telescopic codes," in *Communications (ICC), 2015 IEEE International Conference on*, IEEE, pp. 4505–4510.

[66] Heide, J. (2016) "Composite extension finite fields for distributed storage erasure coding," in *Communications (ICC), 2016 IEEE International Conference on*, IEEE, pp. 1–5.

[67] Geil, O. and D. E. Lucani (2017) "Random Network Coding over Composite Fields," in *International Castle Meeting on Coding Theory and Applications*, Springer, pp. 118–127.

[68] Maymounkov, P., N. J. Harvey, D. S. Lun, et al. (2006) "Methods for efficient network coding," in *Proc. 44th Annual Allerton Conference on Communication, Control, and Computing*, pp. 482–491.

[69] Li, Y., E. Soljanin, and P. Spasojevic (2010) "Collecting coded coupons over overlapping generations," in *Network Coding (NetCod), 2010 IEEE International Symposium on*, IEEE, pp. 1–6.

[70] Silva, D., W. Zeng, and F. R. Kschischang (2009) "Sparse network coding with overlapping classes," in *Network Coding, Theory, and Applications, 2009. NetCod'09. Workshop on*, IEEE, pp. 74–79.

[71] Li, Y., W.-Y. Chan, and S. D. Blostein (2012) "Network coding with unequal size overlapping generations," in *2012 International Symposium on Network Coding (NetCod)*, IEEE, pp. 161–166.

[72] Sorensen, C. W., D. E. Lucani, F. H. Fitzek, and M. Médard (2014) "On-the-fly overlapping of sparse generations: A tunable sparse network coding perspective," in *Vehicular Technology Conference (VTC Fall), 2014 IEEE 80th*, IEEE, pp. 1–5.

[73] Wunderlich, S., F. Gabriel, S. Pandi, and F. H. Fitzek (2017) "We don't need no generation-a practical approach to sliding window RLNC," in *Wireless Days, 2017*, IEEE, pp. 218–223.

[74] Thomos, N. and P. Frossard (2012) "Toward one symbol network coding vectors," *IEEE Communications letters*, **16**(11), pp. 1860–1863.

[75] Gligoroski, D., K. Kralevska, and H. Øverby (2015) "Minimal header overhead for random linear network coding," in *2015 IEEE International Conference on Communication Workshop (ICCW)*, IEEE, pp. 680–685.

[76] Chao, C.-C., C.-C. Chou, and H.-Y. Wei (2010) "Pseudo random network coding design for IEEE 802.16 m enhanced multicast and broadcast service," in *2010 IEEE 71st Vehicular Technology Conference*, IEEE, pp. 1–5.

[77] Khirallah, C., D. Vukobratovic, and J. Thompson (2012) "Performance analysis and energy efficiency of random network coding in LTE-advanced," *IEEE Transactions on Wireless Communications*, **11**(12), pp. 4275–4285.

[78] Tassi, A., C. Khirallah, D. Vukobratović, F. Chiti, J. S. Thompson, and R. Fantacci (2014) "Resource allocation strategies for network-coded video broadcasting services over LTE-advanced," *IEEE Transactions on Vehicular Technology*, **64**(5), pp. 2186–2192.

[79] Cooper, C. (2000) "On the distribution of rank of a random matrix over a finite field," *Random Structures & Algorithms*, **17**(3-4), pp. 197–212.

[80] Boneh, A. and M. Hofri (1997) "The coupon-collector problem revisited—a survey of engineering problems and computational methods," *Stochastic Models*, **13**(1), pp. 39–66.

[81] Li, Y., E. Soljanin, and P. Spasojevic (2011) “Effects of the generation size and overlap on throughput and complexity in randomized linear network coding,” *IEEE Transactions on Information Theory*, **57**(2), pp. 1111–1123.

[82] Trullols-Cruces, O., J. M. Barcelo-Ordinas, and M. Fiore (2011) “Exact decoding probability under random linear network coding,” *IEEE communications letters*, **15**(1), pp. 67–69.

[83] Zhao, X. (2012) “Notes on" exact decoding probability under random linear network coding",” *IEEE communications letters*, **16**(5), pp. 720–721.

[84] Claridge, J. and I. Chatzigeorgiou (2017) “Probability of partially decoding network-coded messages,” *IEEE Communications Letters*, **21**(9), pp. 1945–1948.

[85] Lucani, D. E., M. V. Pedersen, D. Ruano, C. W. Sørensen, F. H. Fitzek, J. Heide, and O. Geil (2014) “Fulcrum network codes: A code for fluid allocation of complexity,” *arXiv preprint arXiv:1404.6620*.

[86] Shrader, B. and N. M. Jones (2009) “Systematic wireless network coding,” in *MILCOM 2009-2009 IEEE Military Communications Conference*, IEEE, pp. 1–7.

[87] Lucani, D. E., M. Médard, and M. Stojanovic (2010) “Systematic network coding for time-division duplexing,” in *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, IEEE, pp. 2403–2407.

[88] Jones, A. L., I. Chatzigeorgiou, and A. Tassi (2015) “Binary systematic network coding for progressive packet decoding,” in *2015 IEEE International Conference on Communications (ICC)*, IEEE, pp. 4499–4504.

[89] Heide, J., M. V. Pedersen, F. H. Fitzek, and T. Larsen (2009) “Network coding for mobile devices-systematic binary random rateless codes,” in *2009 IEEE International Conference on Communications Workshops*, IEEE, pp. 1–6.

[90] Chatzigeorgiou, I. and A. Tassi (2017) “Decoding delay performance of random linear network coding for broadcast,” *IEEE Transactions on Vehicular Technology*, **66**(8), pp. 7050–7060.

[91] Zarei, A., P. Pahlevani, and M. Davoodi (2018) “On the Partial Decoding Delay of Sparse Network Coding,” *IEEE Communications Letters*.

[92] Heide, J., M. V. Pedersen, F. H. Fitzek, and M. Médard (2011) “On code parameters and coding vector representation for practical RLNC,” in *Communications (ICC), 2011 IEEE International Conference on*, IEEE, pp. 1–5.

[93] Sørensen, C. W., A. Paramanathan, J. A. Cabrera, M. V. Pedersen, D. E. Lucani, and F. H. Fitzek (2016) “Leaner and meaner: Network coding in SIMD enabled commercial devices,” in *2016 IEEE Wireless Communications and Networking Conference*, IEEE, pp. 1–6.

[94] Garrido, P., D. E. Lucani, and R. Agüero (2017) "Markov chain model for the decoding probability of sparse network coding," *IEEE Transactions on Communications*, **65**(4), pp. 1675–1685.

[95] Charlap, L. S., H. D. Rees, and D. P. Robbins (1990) "The asymptotic probability that a random biased matrix is invertible," *Discrete Mathematics*, **82**(2), pp. 153–163.

[96] Kahn, J. and J. Komlós (2001) "Singularity probabilities for random matrices over finite fields," *Combinatorics, Probability & Computing*, **10**(2), p. 137.

[97] Feizi, S., D. E. Lucani, C. W. Sørensen, A. Makhdoumi, and M. Médard (2014) "Tunable sparse network coding for multicast networks," in *Network Coding (NetCod), 2014 International Symposium on*, IEEE, pp. 1–6.

[98] Feizi, S., D. E. Lucani, and M. Médard (2012) "Tunable Sparse Network Coding," in *Proc. of the Int. Zurich Seminar on Comm.*, pp. 107–110.

[99] Sorensen, C. W., A. S. Badr, J. A. Cabrera, D. E. Lucani, J. Heide, and F. H. Fitzek (2015) "A practical view on tunable sparse network coding," in *Proceedings of European Wireless 2015; 21th European Wireless Conference*, VDE, pp. 1–6.

[100] Fiandrotti, A., V. Bioglio, M. Grangetto, R. Gaeta, and E. Magli (2014) "Band codes for energy-efficient network coding with application to P2P mobile streaming," *IEEE Transactions on Multimedia*, **16**(2), pp. 521–532.

[101] Li, W., F. Bassi, and M. Kieffer (2016) "Sparse random linear network coding for data compression in WSNs," in *2016 IEEE International Symposium on Information Theory (ISIT)*, IEEE, pp. 2729–2733.

[102] Heide, J., M. V. Pedersen, F. H. Fitzek, and M. Médard (2014) "A perpetual code for network coding," in *Vehicular Technology Conference (VTC Spring), 2014 IEEE 79th*, IEEE, pp. 1–6.

[103] Maymounkov, P. (2011) "Perpetual codes: cache-friendly coding," *Unpublished draft, retieved 2nd of September.*

[104] Pahlevani, P., S. Crisóstomo, and D. E. Lucani (2016) "An analytical model for perpetual network codes in packet erasure channels," in *International Workshop on Multiple Access Communications*, Springer, pp. 126–135.

[105] Birk, Y. and T. Kol (2006) "Coding on demand by an informed source (ISCOD) for efficient broadcast of different supplemental data to caching clients," *IEEE Transactions on Information Theory*, **52**(6), pp. 2825–2830.

[106] El Rouayheb, S., A. Sprintson, and C. Georghiades (2010) "On the index coding problem and its relation to network coding and matroid theory," *IEEE Transactions on Information Theory*, **56**(7), pp. 3187–3195.

[107] Qureshi, J., C. H. Foh, and J. Cai (2012) "Optimal solution for the index coding problem using network coding over GF (2)," in *2012 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, IEEE, pp. 209–217.

[108] Effros, M., S. El Rouayheb, and M. Langberg (2015) "An equivalence between network coding and index coding," *IEEE Transactions on Information Theory*, **61**(5), pp. 2478–2487.

[109] Marcano, N. J. H., J. Heide, D. E. Lucani, and F. H. Fitzek (2015) "On the Overhead of Telescopic Codes in Network Coded Cooperation," in *Vehicular Technology Conference (VTC Fall), 2015 IEEE 82nd*, IEEE, pp. 1–6.

[110] Yang, S. and R. W. Yeung (2011) "Coding for a network coded fountain," in *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, IEEE, pp. 2647–2651.

[111] Mahdaviani, K., M. Ardakani, H. Bagheri, and C. Tellambura (2012) "Gamma codes: A low-overhead linear-complexity network coding solution," in *Network Coding (NetCod), 2012 International Symposium on*, IEEE, pp. 125–130.

[112] Baraniuk, R. G. (2007) "Compressive sensing [lecture notes]," *IEEE signal processing magazine*, **24**(4), pp. 118–121.

[113] Donoho, D. L. (2006) "Compressed sensing," *IEEE Transactions on information theory*, **52**(4), pp. 1289–1306.

[114] Choi, J. W., B. Shim, Y. Ding, B. Rao, and D. I. Kim (2017) "Compressed sensing for wireless communications: Useful tips and tricks," *IEEE Communications Surveys & Tutorials*, **19**(3), pp. 1527–1550.

[115] Strohmer, T. and R. W. Heath Jr (2003) "Grassmannian frames with applications to coding and communication," *Applied and computational harmonic analysis*, **14**(3), pp. 257–275.

[116] Donoho, D. L. and X. Huo (2001) "Uncertainty principles and ideal atomic decomposition," *IEEE transactions on information theory*, **47**(7), pp. 2845–2862.

[117] Obermeier, R. and J. A. Martinez-Lorenzo (2017) "Sensing matrix design via mutual coherence minimization for electromagnetic compressive imaging applications," *IEEE Transactions on Computational Imaging*, **3**(2), pp. 217–229.

[118] Cohen, A., W. Dahmen, and R. DeVore (2009) "Compressed sensing and best k-term approximation," *Journal of the American mathematical society*, **22**(1), pp. 211–231.

[119] Candes, E. J. and T. Tao (2006) "Near-optimal signal recovery from random projections: Universal encoding strategies?" *IEEE transactions on information theory*, **52**(12), pp. 5406–5425.

[120] Cahill, J., X. Chen, and R. Wang (2016) "The gap between the null space property and the restricted isometry property," *Linear Algebra and its Applications*, **501**, pp. 363–375.

[121] Candes, E. and J. Romberg (2007) "Sparsity and incoherence in compressive sampling," *Inverse problems*, **23**(3), p. 969.

[122] Zhang, X. and S. Li (2013) "Compressed Sensing via Dual Frame Based ℓ_1-Analysis With Weibull Matrices," *IEEE Signal Processing Letters*, **20**(3), pp. 265–268.

[123] Applebaum, L., S. D. Howard, S. Searle, and R. Calderbank (2009) "Chirp sensing codes: Deterministic compressed sensing measurements for fast recovery," *Applied and Computational Harmonic Analysis*, **26**(2), pp. 283–290.

[124] Zhang, Z., Y. Xu, J. Yang, X. Li, and D. Zhang (2015) "A survey of sparse representation: algorithms and applications," *IEEE access*, **3**, pp. 490–530.

[125] Candes, E. and J. Romberg (2005) "l1-magic: Recovery of sparse signals via convex programming," *URL: www. acm. caltech. edu/l1magic/downloads/l1magic. pdf*, **4**, p. 14.

[126] Figueiredo, M. A., R. D. Nowak, and S. J. Wright (2007) "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *IEEE Journal of selected topics in signal processing*, **1**(4), pp. 586–597.

[127] Tibshirani, R. (1996) "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, **58**(1), pp. 267–288.

[128] Chen, S. S., D. L. Donoho, and M. A. Saunders (2001) "Atomic decomposition by basis pursuit," *SIAM review*, **43**(1), pp. 129–159.

[129] Donoho, D. L., A. Maleki, and A. Montanari (2009) "Message-passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences*, **106**(45), pp. 18914–18919.

[130] Tropp, J. A. and A. C. Gilbert (2007) "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on information theory*, **53**(12), pp. 4655–4666.

[131] Needell, D. and R. Vershynin (2010) "Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit," *IEEE Journal of selected topics in signal processing*, **4**(2), pp. 310–316.

[132] Needell, D. and J. A. Tropp (2009) "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, **26**(3), pp. 301–321.

[133] Dai, W. and O. Milenkovic (2009) "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Transactions on Information Theory*, **55**(5), pp. 2230–2249.

[134] Bredies, K. and D. A. Lorenz (2008) "Linear convergence of iterative soft-thresholding," *Journal of Fourier Analysis and Applications*, **14**(5-6), pp. 813–837.

[135] Blumensath, T. and M. E. Davies (2009) "Iterative hard thresholding for compressed sensing," *Applied and computational harmonic analysis*, **27**(3), pp. 265–274.

[136] Krzakala, F., M. Mézard, F. Sausset, Y. Sun, and L. Zdeborová (2012) "Statistical-physics-based reconstruction in compressed sensing," *Physical Review X*, **2**(2), p. 021005.

[137] Mohimani, H., M. Babaie-Zadeh, and C. Jutten (2009) "A Fast Approach for Overcomplete Sparse Decomposition Based on Smoothed l0 Norm," *IEEE Transactions on Signal Processing*, **57**(1), pp. 289–301.

[138] Torre, R., T. Doan, and H. Salah (2021) "Mobile edge cloud," in *Computing in Communication Networks*, Elsevier, pp. 77–91.

[139] Kutyniok, G. and W.-Q. Lim (2011) "Compactly supported shearlets are optimally sparse," *Journal of Approximation Theory*, **163**(11), pp. 1564–1589.

[140] Qin, Z., J. Fan, Y. Liu, Y. Gao, and G. Y. Li (2018) "Sparse representation for wireless communications: A compressive sensing approach," *IEEE Signal Processing Magazine*, **35**(3), pp. 40–58.

[141] Marcellin, M. W., M. J. Gormish, A. Bilgin, and M. P. Boliek (2000) "An overview of JPEG-2000," in *Proceedings DCC 2000. Data Compression Conference*, IEEE, pp. 523–541.

[142] Engan, K., S. O. Aase, and J. H. Husoy (1999) "Method of optimal directions for frame design," in *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258)*, vol. 5, IEEE, pp. 2443–2446.

[143] Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011) "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, **12**, pp. 2825–2830.

[144] Sarvotham, S., D. Baron, M. Wakin, M. F. Duarte, and R. G. Baraniuk (2005) "Distributed compressed sensing of jointly sparse signals," in *Asilomar conference on signals, systems, and computers*, pp. 1537–1541.

[145] Qiao, W., B. Liu, and C. W. Chen (2012) "JSM-2 based joint ECG compressed sensing with partially known support establishment," in *2012 IEEE 14th International Conference on e-Health Networking, Applications and Services (Healthcom)*, IEEE, pp. 435–438.

[146] Wang, B., Y. Ge, C. He, Y. Wu, and Z. Zhu (2019) "Study on communication channel estimation by improved SOMP based on distributed compressed sensing," *EURASIP Journal on Wireless Communications and Networking*, **2019**(1), p. 121.

[147] Tropp, J. A., A. C. Gilbert, and M. J. Strauss (2005) "Simultaneous sparse approximation via greedy pursuit," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP).*

[148] Baron, D., M. F. Duarte, M. B. Wakin, S. Sarvotham, and R. G. Baraniuk (2009) "Distributed compressive sensing," *arXiv preprint arXiv:0901.3403.*

[149] Comon, P. and C. Jutten (2010) *Handbook of Blind Source Separation: Independent component analysis and applications*, Academic press.

[150] Wu, H., Y. Shen, J. Zhang, I. A. Tsokalo, H. Salah, and H. P. Frank Fitzek (2020) "Component-Dependent Independent Component Analysis for Time-Sensitive Applications," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pp. 1–6.

[151] Gemmeke, J. F., D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter (2017) "Audio Set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 776–780.

[152] CAIDA, "Packet size distribution comparison between internet links in 1998 and 2008," .
URL `http://www.caida.org/research/traffic-analysis/pkt_size_distribution/graphs.xml`

[153] "The CAIDA Anonymized Internet Traces 2015 Dataset," .
URL `http://www.caida.org/data/passive/passive_2015_dataset.xml`

[154] Seeling, P. and M. Reisslein (2012) "Video Transport Evaluation With H.264 Video Traces," *Communications Surveys Tutorials, IEEE*, **14**(4), pp. 1142–1165.

[155] Sinha, R., C. Papadopoulos, and J. Heidemann (2007) "Internet packet size distributions: Some observations," *USC/Information Sciences Institute, Tech. Rep. ISI-TR-2007-643*.

[156] CAIDA, "Packet size distribution comparison between Internet links in 1998 and 2008," `https://www.caida.org/research/traffic-analysis/pkt_size_distribution/graphs.xml`, current: June 2018.

[157] (2015), "Cisco visual networking index: Forecast and methodology, 2014-2019 White Paper," .
URL `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html`

[158] Kishore, M. and Y. Liang (2006) "An empirical study on renegotiated CBR for VBR video services based on network testbed," *IEEE Transactions on Broadcasting*, **52**(3), pp. 362–367.

[159] Lakshman, T., A. Ortega, and A. R. Reibman (1998) "VBR video: Tradeoffs and potentials," *Proceedings of the IEEE*, **86**(5), pp. 952–973.

[160] Rerabek, M. and T. Ebrahimi (2014) "Comparison of compression efficiency between HEVC/H.265 and VP9 based on subjective assessments," in *Proc. SPIE Appl. of Dig. Image Proc.*, vol. 9217, pp. 1–13.

[161] Seeling, P. and M. Reisslein (2014) "Video traffic characteristics of modern encoding standards: H.264/AVC with SVC and MVC extensions and H.265/HEVC," *The Scientific World J.*, **2014**(189481), pp. 1–16.

[162] Tanwir, S. and H. Perros (2013) "A survey of VBR video traffic models," *IEEE Commun. Surv. & Tut.*, **15**(4), pp. 1778–1802.

[163] Yoo, S.-J. (2002) "Efficient traffic prediction scheme for real-time VBR MPEG video transmission over high-speed networks," *IEEE Transactions on Broadcasting*, **48**(1), pp. 10–18.

[164] Pulipaka, A., P. Seeling, M. Reisslein, and L. J. Karam (2013) "Traffic and statistical multiplexing characterization of 3-D video representation formats," *Broadcasting, IEEE Transactions on*, **59**(2), pp. 382–389.

[165] der Auwera, G. V., P. T. David, and M. Reisslein (2008) "Traffic and Quality Characterization of Single-Layer Video Streams Encoded with the H.264/MPEG-4 Advanced Video Coding Standard and Scalable Video Coding Extension," *IEEE Transactions on Broadcasting*, **54**(3), pp. 698–718.

[166] der Auwera, G. V. and M. Reisslein (2009) "Implications of Smoothing on Statistical Multiplexing of H.264/AVC and SVC Video Streams," *IEEE Transactions on Broadcasting*, **55**(3), pp. 541–558.

[167] Heide, J., M. Pedersen, F. Fitzek, and M. Médard (2011) "On Code Parameters and Coding Vector Representation for Practical RLNC," in *Proc. of the IEEE International Conference on Communications (ICC'11)*, pp. 1 –5.

[168] Li, Y., E. Soljanin, and P. Spasojevic (2011) "Effects of the Generation Size and Overlap on Throughput and Complexity in Randomized Linear Network Coding," *IEEE Trans. on Info. Theory*, **57**(2), pp. 1111–1123.

[169] Compta, P. T., F. H. P. Fitzek, and D. E. Lucani (2015) "Network coding is the 5G Key Enabling Technology: effects and strategies to manage heterogeneous packet lengths," *Trans. Emerging Telecommunications Technologies*, **26**(1), pp. 46–55.

[170] Johnson, D. S. (1973) *Near-optimal bin packing algorithms*, Ph.D. thesis, Massachusetts Institute of Technology.

[171] Pedersen, M. V., J. Heide, and F. H. Fitzek (2011) "Kodo: An open and research oriented network coding library," in *International Conference on Research in Networking*, Springer, pp. 145–152.

[172] Sinha, R., C. Papadopoulos, and J. Heidemann (2007) *Internet Packet Size Distributions: Some Observations, Tech. Rep. ISI-TR-2007-643*, USC/Information Sciences Institute.

[173] Heide, J., M. V. Pedersen, and F. H. Fitzek (2011) "Decoding algorithms for random linear network codes," in *Proc. Int. Conf. on Research in Networking*, Springer, Heidelberg, Germany, pp. 129–136.

[174] Karzand, M., D. J. Leith, J. Cloud, and M. Medard (2017) "Design of FEC for low delay in 5G," *IEEE Journal on Selected Areas in Communications*, **35**(8), pp. 1783–1793.

[175] Chatzigeorgiou, I. and A. Tassi (2017) "Decoding delay performance of random linear network coding for broadcast," *IEEE Transactions on Vehicular Technology*, **66**(8), pp. 7050–7060.

[176] Heide, J., M. V. Pedersen, and F. H. Fitzek (2011) "Decoding algorithms for random linear network codes," in *International Conference on Research in Networking*, Springer, pp. 129–136.

[177] Acevedo, J., R. Scheffel, S. Wunderlich, M. Hasler, S. Pandi, J. Cabrera, F. Fitzek, G. Fettweis, and M. Reisslein (2018) "Hardware Acceleration for RLNC: A Case Study Based on the Xtensa Processor with the Tensilica Instruction-Set Extension," *Electronics*, **7**(9), pp. 180.1–180.22.

[178] Shin, H. and J.-S. Park (2017) "Optimizing random network coding for multimedia content distribution over smartphones," *Multimedia Tools and Applications*, **76**, pp. 19379–19395.

[179] Wunderlich, S., J. Cabrera, F. H. Fitzek, and M. Reisslein (2017) "Network Coding in Heterogeneous Multicore IoT Nodes with DAG Scheduling of Parallel Matrix Block Operations," *IEEE Internet of Things Journal*, **4**(4), pp. 917–933.

[180] Pelletier, G. and K. Sandlund (1997) "RObust Header Compression Version 2 (ROHCv2): Profiles for RTP, UDP, IP, ESP and UDP-Lite," *Request for Comments 5225*.

[181] Tomoskozi, M., P. Seeling, P. Ekler, and F. H. Fitzek (21-25 May 2017) "Robust Header Compression version 2 power consumption on Android devices via tunnelling," *2017 IEEE International Conference on Communications Workshops*.

[182] Tömösközi, M., P. Seeling, P. Ekler, and F. H. P. Fitzek (2016) "Performance Evaluation of Network Header Compression Schemes for UDP, RTP and TCP," *Periodica Polytechnica Electrical Engineering and Computer Science*, p. Online First paper 8958.

[183] Tömösközi, M., P. Seeling, P. Ekler, and F. H. P. Fitzek (2016) "Efficiency Gain for RoHC Compressor Implementations with Dynamic Configuration," in *VTC2016-Fall Workshop on Cellular Internet of Things - Emerging Trends and Enabling Technologies*.

[184] Tomoskozi, M., P. Seeling, P. Ekler, and F. H. P. Fitzek (2016) "Regression Model Building and Efficiency Prediction of RoHCv2 Compressor Implementations for VoIP," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6.

[185] Barros, J., R. A. Costa, D. Munaretto, and J. Widmer (2009) "Effective delay control in online network coding," in *INFOCOM 2009, IEEE*, IEEE, pp. 208–216.

[186] Lucani, D. E., M. Médard, and M. Stojanovic (2010) "Systematic network coding for time-division duplexing," .

[187] ns 3 (2012), "ns-3 a discrete-event network simulator for internet systems," .
URL `https://www.nsnam.org/`

[188] Pedersen, M., J. Heide, F.H.P.Fitzek, and T. Larsen (2009) "PictureViewer - A Mobile Application using Network Coding," in *European Wireless 2009*, Aalborg, Denmark.

[189] Baron, D., M. B. Wakin, M. F. Duarte, S. Sarvotham, and R. G. Baraniuk (2005) "Distributed compressed sensing," .

[190] Katti, S., S. Shintre, S. Jaggi, D. Katabi, and M. Medard (2007) "Real network codes," in *Proc. Forty-Fifth Annual Allerton Conference*, pp. 389–395.

[191] Dey, B., S. Katti, S. Jaggi, D. Katabi, M. Médard, and S. Shintre (2008) "Real and complex network codes: Promises and challenges," in *Network Coding, Theory and Applications, 2008. NetCod 2008. Fourth Workshop on*, IEEE, pp. 1–6.

[192] Draper, S. C. and S. Malekpour (2009) "Compressed sensing over finite fields," in *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, IEEE, pp. 669–673.

[193] Seong, J.-T. and H.-N. Lee (2013) "Necessary and sufficient conditions for recovery of sparse signals over finite fields," *IEEE Communications Letters*, **17**(10), pp. 1976–1979.

[194] ——— (2012) "On the compressed measurements over finite fields: Sparse or dense sampling," *arXiv preprint arXiv:1211.5207*.

[195] Bioglio, V., T. Bianchi, and E. Magli (2014) "Secure compressed sensing over finite fields," in *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*, IEEE, pp. 191–196.

[196] Bioglio, V., G. Coluccia, and E. Magli (2014) "Sparse image recovery using compressed sensing over finite alphabets," in *2014 IEEE International Conference on Image Processing (ICIP)*, IEEE, pp. 1287–1291.

[197] Xie, D., H. Peng, L. Li, and Y. Yang (2017) "An efficient privacy-preserving scheme for secure network coding based on compressed sensing," *AEU-International Journal of Electronics and Communications*, **79**, pp. 33–42.

[198] Dehghan, H., I. Lambadaris, and C.-H. Lung (2012) "Network coding based wideband compressed spectrum sensing," in *2012 IEEE International Conference on Communications (ICC)*, IEEE, pp. 1405–1409.

[199] Yang, X., X. Tao, E. Dutkiewicz, X. Huang, Y. J. Guo, and Q. Cui (2013) "Energy-efficient distributed data storage for wireless sensor networks based on compressed sensing and network coding," *IEEE Transactions on Wireless Communications*, **12**(10), pp. 5087–5099.

[200] Nabaee, M. and F. Labeau (2012) "Quantized network coding for sparse messages," in *2012 IEEE Statistical Signal Processing Workshop (SSP)*, IEEE, pp. 828–831.

[201] ——— (2014) "Quantized network coding for correlated sources," *EURASIP Journal on Wireless Communications and Networking*, **2014**(1), pp. 1–17.

[202] Katti, S., S. Gollakota, and D. Katabi (2007) "Embracing wireless interference: Analog network coding," *ACM SIGCOMM Computer Communication Review*, **37**(4), pp. 397–408.

[203] Fan, N., Y. Liu, and L. Zhang (2013) "Combination of analog network coding and compressed sensing in clustered chain-type topology," in *Communication Technology (ICCT), 2013 15th IEEE International Conference on*, IEEE, pp. 797–801.

[204] Chen, S., M. Wu, K. Wang, and Z. Sun (2014) "Compressive network coding for error control in wireless sensor networks," *Wireless networks*, **20**(8), pp. 2605–2615.

[205] Ebrahimi, D. and C. Assi (2015) "On the benefits of network coding to compressive data gathering in wireless sensor networks," in *Sensing, Communication, and Networking (SECON), 2015 12th Annual IEEE International Conference on*, IEEE, pp. 55–63.

[206] Liu, X., K. Cao, F. Han, and P. Cull (2017) "A Practical Joint Network-Compressed Coding Scheme for Energy-Efficient Data Gathering in Cooperative Wireless Sensor Networks," in *Computational Science and Engineering (CSE) and Embedded and Ubiquitous Computing (EUC), 2017 IEEE International Conference on*, vol. 2, IEEE, pp. 70–76.

[207] Nguyen, N., D. L. Jones, and S. Krishnamurthy (2010) "Netcompress: Coupling network coding and compressed sensing for efficient data communication in wireless sensor networks," in *2010 IEEE Workshop On Signal Processing Systems*, pp. 356–361.

[208] Chou, P. A., Y. Wu, and K. Jain (2003) "Practical network coding," in *Allerton Conference on Communication, Control, and Computing*, pp. 40–49, invited paper.

[209] Feizi, S., M. Médard, and M. Effros (2010) "Compressive sensing over networks," in *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, IEEE, pp. 1129–1136.

[210] Feizi, S. and M. Médard (2011) "A power efficient sensing/communication scheme: Joint source-channel-network coding by using compressive sensing," in *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, pp. 1048–1054.

[211] Luo, C., J. Sun, and F. Wu (2011) "Compressive network coding for approximate sensor data gathering," in *2011 IEEE Global Telecommunications Conference-GLOBECOM 2011*, IEEE, pp. 1–6.

[212] Das, A. K. and S. Vishwanath (2013) "On finite alphabet compressive sensing," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, IEEE, pp. 5890–5894.

[213] Kwon, M., H. Park, and P. Frossard (2014) "Compressed network coding: Overcome all-or-nothing problem in finite fields," in *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, IEEE, pp. 2851–2856.

[214] Gankhuyag, G., E. Hong, and Y. Choe (2017) "Sparse recovery using sparse sensing matrix based finite field optimization in network coding," *IEICE TRANSACTIONS on Information and Systems*, **100**(2), pp. 375–378.

[215] Li, W., F. Bassi, and M. Kieffer (2014) "Robust Bayesian compressed sensing over finite fields: asymptotic performance analysis," *arXiv preprint arXiv:1401.4313*.

[216] Li, Y., J. Li, Z. Lin, Y. Li, and B. Vucetic (2014) "Compressive soft forwarding in network-coded multiple-access relay channels," *IEEE Transactions on Vehicular Technology*, **64**(5), pp. 2138–2144.

[217] Raskutti, G., M. J. Wainwright, and B. Yu (2010) "Restricted eigenvalue properties for correlated Gaussian designs," *The Journal of Machine Learning Research*, **11**, pp. 2241–2259.

[218] Gilbert, A. and P. Indyk (2010) "Sparse recovery using sparse matrices," *Proceedings of the IEEE*, **98**(6), pp. 937–947.

[219] Shen, W., G. Han, L. Shu, J. J. Rodrigues, and N. Chilamkurti (2011) "A new energy prediction approach for intrusion detection in cluster-based wireless sensor networks," in *International Conference on Green Communications and Networking*, Springer, pp. 1–12.

[220] Cao, D.-Y., K. Yu, S.-G. Zhuo, Y.-H. Hu, and Z. Wang (2016) "On the implementation of compressive sensing on wireless sensor network," in *Internet-of-Things Design and Implementation (IoTDI), 2016 IEEE First International Conference on*, IEEE, pp. 229–234.

[221] Wiederhold, B. K. (2020), "Connecting through technology during the coronavirus disease 2019 pandemic: Avoiding "Zoom Fatigue"," .

[222] Fu, X., S. Yang, and Z. Xiao (2020) "Decoding and Repair Schemes for Shift-XOR Regenerating Codes," *IEEE Transactions on Information Theory*, **66**(12), pp. 7371–7386.

[223] Wunderlich, S., F. Gabriel, S. Pandi, F. H. Fitzek, and M. Reisslein (2017) "Caterpillar RLNC (CRLNC): A Practical Finite Sliding Window RLNC Approach," *IEEE Access*, **5**, pp. 20183–20197.

[224] Sundararajan, J. K., D. Shah, M. Medard, and P. Sadeghi (2017) "Feedback-Based Online Network Coding," *IEEE Trans. on Information Theory*, **63**(10), pp. 6628–6649.

[225] Boufounos, P. T. and R. G. Baraniuk (2008) "1-bit compressive sensing," in *2008 42nd Annual Conference on Information Sciences and Systems*, IEEE, pp. 16–21.

[226] Koep, N. and R. Mathar (2017) "Binary iterative hard thresholding for frequency-sparse signal recovery," in *WSA 2017; 21th International ITG Workshop on Smart Antennas*, VDE, pp. 1–7.

[227] LeCun, Y., Y. Bengio, and G. Hinton (2015) "Deep learning," *nature*, **521**(7553), pp. 436–444.

[228] Khan, L. U., I. Yaqoob, M. Imran, Z. Han, and C. S. Hong (2020) "6G wireless systems: A vision, architectural elements, and future directions," *IEEE Access*, **8**, pp. 147029–147044.

[229] Tariq, F., M. R. Khandaker, K.-K. Wong, M. A. Imran, M. Bennis, and M. Debbah (2020) "A speculative study on 6G," *IEEE Wireless Communications*, **27**(4), pp. 118–125.

[230] Madden, S., "Intel lab data," .
URL `http://db.csail.mit.edu/labdata/labdata.html`