# Network Coding Using Superregular Matrices For Robust Real-Time Streaming

Hansen, Jonas

*DOI (link to publication from Publisher):*
10.5278/vbn.phd.tech.00035

*Publication date:*
2018

*Document Version*
Publisher's PDF, also known as Version of record

# NETWORK CODING USING SUPERREGULAR MATRICES FOR ROBUST REAL-TIME STREAMING

BY
JONAS HANSEN

DISSERTATION SUBMITTED 2017

AALBORG UNIVERSITY
DENMARK

# Network Coding Using Superregular Matrices For Robust Real-Time Streaming

Ph.D. Dissertation

Jonas Hansen

Dissertation submitted December, 2017

| | |
|---|---|
| Dissertation submitted: | December, 2017 |
| PhD supervisor: | Prof. Jan Østergaard,<br>Aalborg University |
| Industrial PhD Supervisor: | B.Sc.EE John Hammer Madsen,<br>Bang & Olufsen a/s |
| Industrial PhD Co-supervisor: | B.Sc.EE Johnny Kudahl,<br>Bang & Olufsen a/s |
| PhD committee: | Associate Professor Jimmy Jessen Nielsen (chair.)<br>Aalborg University |
| | Professor Lars K. Rasmussen<br>KTH Royal Institute of Technology |
| | Professor Emina Soljanin<br>Rutgers School of Engineering |
| PhD Series: | Technical Faculty of IT and Design,<br>Aalborg University |
| Department: | Department of Electronic Systems |

ISSN (online): 2446-1628
ISBN (online): 978-87-7210-121-7

Published by:
Aalborg University Press
Langagervej 2
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk



Printed in Denmark by Rosendahls, 2018

This thesis has been typeset using LaTeX 2ε.

# Curriculum Vitae

Jonas Hansen

Jonas Hansen was an industrial Ph.D. student at Bang & Olufsen a/s, Struer, Denmark and Aalborg University, Aalborg, Denmark working on network coding, code design and wireless distribution protocols for audio streaming. He received his B.S. degree in electrical engineering from Aalborg University in 2012. In 2013 he was a visiting student at Massachusetts Institute of Technology's Research Laboratory of Electronics, where he collaborated with the Network Coding and Reliable Communications Group. He was a student in the Elite Masters Programme in Wireless Communication at Aalborg University, where he in 2014 graduated with *cum laude*. From 2006 to 2009 he was an apprentice at Bang & Olufsen a/s where he designed and developed software and hardware for both production and user opinion tests, and in 2009 he graduated as an Electronics Technician. His research interests include network coding for wireless communication and multimedia transmission with an emphasis on low-latency traffic and applications.

Curriculum Vitae

# Abstract

This thesis is about low delay wireless media streaming. Since streaming of audio and video has become a part of our everyday lives, this work seeks to improve the quality of wireless audio streaming. Moreover, the main topic of this thesis is *erasure correcting codes*, that is, codes that are used to correct the erasures of entire packets in wireless networks. These codes may be used to make low delay wireless media streaming more robust, i.e., more tolerant towards erasures caused by errors somewhere along the network path.

In order for erasure correcting codes to be beneficial for low delay streaming, they should optimize the encoding delay, decoding delay, and loss probability. In this work we study codes with a lower triangular structure, which minimizes the encoding delay, as opposed to dense codes, e.g., Reed-Solomon codes, where the source must collect all input symbols before the encoding can take place. The lower triangular structure has also proven favourable with respect to decoding delay, since packets can be decoded on-the-fly. The symbol loss probability is then investigated for these lower triangular codes. The result of this investigation is two-fold. First, a framework for determining the symbol loss probability for any static linear erasure correcting code is composed. Second, in order for a systematic lower triangular code to have optimal decoding capabilities, the redundancy part of the encoding matrix must be superregular. Moreover, a code is said to have optimal decoding capabilities if and only if no other code with the same structure of the encoding matrix is able to produce a lower symbol loss probability over any erasure channel.

Using these erasure correcting codes on modern platforms, such as loudspeakers, requires an efficient software implementation. To this end, this thesis documents a high-performance software library for encoding and decoding linear erasure correcting codes. This software library has proven very valuable for this research project, as it enabled large simulation campaigns and allowed for implementation of a real-time multi-node wireless testbed.

Overall the work documented in this thesis contributes to:

1. The construction of superregular lower triangular Toeplitz matrices over binary extension fields.

2. Analysis and evaluation of the performance, in terms of symbol loss probability and delay, for both block and convolutional codes.

3. A flexible and extendable high performance software library for linear erasure correcting block and convolutional codes.

The thesis is divided into two parts. The first part provides a detailed theoretical background for the thesis while also providing an overview of the state-of-the-art. The second part consists of five academic papers, which in detail describe the research undertaken throughout this project.

# Resumé

Denne afhandling drejer sig om streaming af lyd og video med lav forsinkelse. Da streaming af lyd og video er blevet en del af dagligdagen, forsøger dette projekt at forbedre kvaliteten af trådløs streaming af især lyd. Hoved emnet for denne afhandling er derfor *fejl korrigerende koder*, altså, koder der bruges til at korrigere for de pakke tab der forekommer på trådløse netværk. Disse koder kan bruges til at gøre streaming af lyd og video med lav forsinkelse mere robust, og dermed mere tolerant i forhold til pakke tab forsaget af fejl i netværket.

For at fejl korrigerende koder kan være brugbare for streaming af lyd og video med lav forsinkelse, skal de i nogen grad optimere indkodningsforsinkelsen, dekodningsforsinkelsen og tabs sandsynligheden. I dette projekt har vi studeret koder med en lavere trekantet struktur, som minimere indkodningsforsinkelsen, i modsætning til tætte koder, f.eks. Reed-Solomon koder, hvor afsenderen skal samle alle input symbolerne før kodningen kan begynde. Den lavere trekants struktur har også vist sig at være favorabel med hensyn til dekodningsforsinkelsen, da pakkerne kan dekodes løbende. Derefter er symbol tabs sandsynligheden blevet undersøgt for disse lavere trekantede koder. Der er to resultater af denne undersøgelse. For det første er der udarbejdet en teoretisk ramme som kan bruges til at bestemme symbol tabs sandsynlighed for enhver statisk lineær fejl korrigerende kode. For det andet, for at en systematisk lavere trekantet kode kan have optimale dekodnings betingelser, skal redundansdelen af kodningsmatricen være superregulær. Ydermere, en kode siges at have optimale dekodnings betingelser hvis og kun hvis der ikke findes en anden kode med samme kodnings matrice struktur der er i stand til at producere en lavere symbol tabs sandsynlighed over enhver tabs kanal.

Brug af disse fejl korrigerende koder på moderne platforme, såsom højttalere, kræver en effektiv softwareimplementering. Til dette formål dokumenterer denne afhandling et højtydende softwarebibliotek til kodning og afkodning af lineære fejl korrigerende koder. Dette softwarebibliotek har vist sig meget værdifuldt for dette forskningsprojekt, både fordi det har muliggjort store simuleringskampagner, og da det tillod implementering af en realtids multinode trådløs forsøgsopstilling.

Samlet set bidrager arbejdet i denne afhandling til:

1. Konstruktionen af superregulære lavere trekantede Toeplitz matricer over et hvert endeligt legeme med karakteristik 2.

2. Analyse og evaluering af ydeevnen, hvad angår symboltab sandsynlighed og forsinkelse, for både blok og foldnings koder.

3. Et fleksibelt og udvideligt højtydende softwarebibliotek til lineær fejl korrigerende blok og foldnings koder.

Afhandlingen er opdelt i to dele. Den første del giver detaljeret baggrund stof for afhandlingen, samtidig med at der gives et overblik over state-of-the-art. Anden del består af fem videnskabelige artikler, som i detaljer beskriver den forskning der er gennemført i løbet af projektet.

# Contents

# Preface

This PhD thesis presents a selection of papers which encompass the research topics and directions that were investigated throughout my time as a PhD student at Bang & Olufsen a/s as well as at the Department of Electronic Systems at Aalborg University, from December 2014 through December 2017. This thesis is submitted to the Technical Doctoral School of IT and Design in partial fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis includes 5 selected publications and was prepared under the supervision of Professor Jan Østergaard, DSP Audio Specialist John Hammer Madsen, and Wireless Technology Specialist Johnny Kudahl. This work was partly financed by the Danish National Innovation Foundation, Grant No. 4135-00131B.

Paper A presents results on software implementations of network coding and experiments with bursty wireless erasure channels. What inspired Paper B was the finding that superregular matrices could only be constructed over very large prime fields, given that for practical purposes the best family of fields is binary extension fields, since they may fit easily into the memory of computer systems. In Paper C superregular matrices are used to construct convolutional codes with optimum distance profile.

The following comment from Reviewer 2 of Paper B, regarding rate $1/2$ lower triangular block codes, started the research efforts that led to Papers D and E.

> "*A rate $1/2$ MDS convolutional with the same dimensions will recover from more losses.*"

These papers cover an experimental investigation of the symbol loss probability for random based rate $1/2$ codes given strict delay constraints. One of the findings in those papers is that lower triangular block codes are generally superior to both convolutional and dense codes when used in networks with recoding.

Jonas Hansen
Aalborg University, Monday 25$^{\text{th}}$ December, 2017

# Acknowledgments

First and foremost I would like to thank my supervisors Professor Jan Østergaard, John Hammer Madsen, and Johnny Kudahl. You helped me to make a succes of this project, and guided me through the tough times. I am very glad that we completed this journey together, and I am proud that you were by my side along the way.

I feel very grateful for having had such inspiring colleagues over the three years. Thank you Sven Shepstone, Neo Kaplanis, Martin Møller, Samuel Moulin, Miklas Strøm Kristoffersen, Klaus Kristensen, Pablo Martinez-Nuevo, and Kashmiri Stec. And a special thank you to Søren Bech, for believing in the project idea I suggested almost four years ago.

Finally, to my family and friends, thank you for your continued support. I look forward to be able to spend more time with all of you. In particular thanks to my wife – you are the best.

The work included in this PhD thesis is dedicated to my wife and our kids.

> "*Research is formalized curiosity. It is poking and prying with a purpose.*"
>
> — Zora Neale Hurston, *Dust Tracks on a Road*

# Abbreviations

| | |
|---|---|
| ARQ | Automatic repeat request |
| EGM | Extended Gilbert model |
| FEC | Forward erasure correction |
| MDS | Maximum distance separable |
| ODP | Optimum distance Profile |
| RLNC | Random linear network coding |
| SIMD | Single instruction multiple data |
| SNR | Signal-to-noise ratio |
| WNIC | Wireless network interface card |

# Part I

# Background

# Introduction

## 1 Research objective

The scientific purpose of this project is the analysis, design, implementation, and evaluation of distribution of lossless high-resolution audio with low delay in a wireless one-to-many scenario in the presence of typical in-home wireless network topologies and channel imperfections using off-the-shelf wireless devices. We follow a top-down approach to describe the research documented in this thesis. That is, we start with *Why* was this research undertaken? This is followed by the statement: *How* do we achieve the desired result? Finally, we consider, *What* has been done? These three elements are described in more detail in the coming paragraphs, and are also depicted in Fig. 1.

We wish to distribute lossless high-resolution audio with low delay wirelessly to a number of loudspeakers. However, there are several limitations that we need to overcome, such as limited bandwidth, random packet erasures, stochastic network delay, and limited computational power in the devices. That is, we want to improve the sound quality, by: 1) improving the audio resolution, 2) having a lossless transmission, 3) playing back from a number of wireless loudspeakers simultaneously. This constitutes the *Why*.

Erasure correcting codes allows for a trade-off between delay, bandwidth, and loss probability. It follows easily that, for media streaming the delay should be minimized, but not to the extent that the bandwidth requirements are too high or that the losses are too frequent. Thus, the contributions with respect to the *How* lies in an investigation of the relationship between these three metrics.

Some form of forward erasure correcting code is generally required in order to reliably deliver high-resolution audio to a number of loudspeakers with low delay. Without a forward erasure correcting code the loudspeakers and the audio master have to rely entirely on either positive or negative acknowledgements, in order to achieve lossless delivery of the audio content. This is, however, generally not desired when the number of loudspeakers increases, since the acknowledgements may reduce the available bandwidth. Furthermore, the computational complexity of an erasure correcting code should not exceed the capacity of modern embedded devices, for which the codes are intended to be

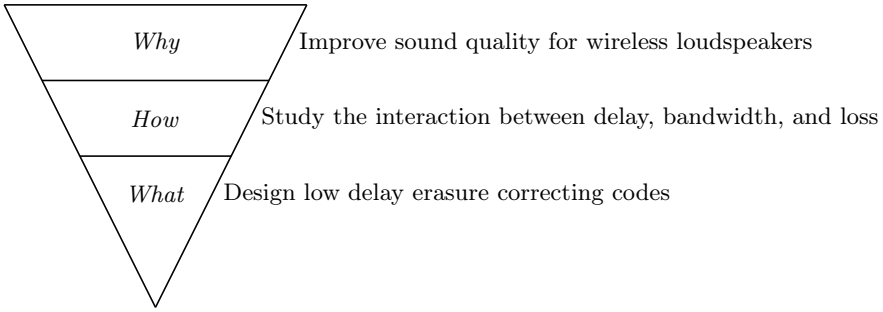



**Fig. 1:** The motivation behind the research documented within this thesis.

used. For those reasons, this project revolves around erasure correcting codes that perform well with respect to low delay media streaming. That is, the *What* is about the design, analysis, and evaluation of erasure correcting codes.

# 2 Commercial Perspectives

Bang & Olufsen a/s may utilize the results of the research conducted throughout this project to develop loudspeakers and television sets capable of streaming lossless high-resolution audio to multiple receivers with low delay. One of the commercial targets of this project is to completely remove wired signal-connections to Bang & Olufsen a/s loudspeakers when connected to a Bang & Olufsen a/s audio master or television set. This puts a large emphasize on the need for erasure correcting codes in software which are capable of high performance encoding and decoding of the audio data on embedded devices. Furthermore, it also requires research into the areas of low delay erasure correcting codes. That is, the codes should be with low delay in terms of end-to-end symbol delay and with respect to encoding and decoding.

By using an off-the-shelf wireless technology such as Wi-Fi, in contrast to the currently implemented proprietary wireless system, it is possible to reduce both production and development costs. However, the use of off-the-shelf wireless technologies is generally not unproblematic. The reason for this is that a general purpose wireless technology has not been developed for a very specific use-case in mind, whereas, the current proprietary wireless system was designed exactly for low delay audio streaming through a long and costly development process. Thus, from a commercial perspective, it is highly desired to conduct the necessary research required to enable streaming of high-resolution audio over a general purpose wireless technology with low delay. This will include research into erasure correcting codes and their interplay with the underlying wireless technology.

# 3 Thesis Structure

The structure of this thesis is as follows. The following section covers the contributions of the thesis. The remainder of Part I provides an extended summary of the background for the thesis, description of state-of-the-art, and conclusions. Part II contains the following five papers:

A) J. Hansen, J. Østergaard, J. Kudahl, J. Madsen. "Network Coding on Embedded Platforms for Wireless Networks". Technical report, December 2017.

B) J. Hansen, J. Østergaard, J. Kudahl, J. Madsen. "Superregular Lower Triangular Toeplitz Matrices for Low Delay Wireless Streaming". Published in the *IEEE Transactions on Communications* Vol. 65, no. 9, pp. 4027–4038, September 2017.

C) J. Hansen, J. Østergaard, J. Kudahl, J. Madsen. "On Superregular Matrices and Convolutional Codes with Finite Decoder Memory". Submitted to the *IEEE 87th Vehicular Technology Conference*, pp. 1–5, June 2018.

D) J. Hansen, J. Østergaard, J. Kudahl, J. Madsen. "When are Erasure Correcting Block Codes Better than Convolutional Codes in a Multi-hop Network?". Published in in the proceedings of *The 11th International Conference on Signal Processing and Communication Systems*, pp. 1–5, December 2017.

E) J. Hansen, J. Østergaard, J. Kudahl, J. Madsen. "Sequential use of Block Codes and Convolutional Codes in a Real-Time Multi-hop Network". Submitted to the *IEEE 87th Vehicular Technology Conference*, pp. 1–5, June 2018.

In addition to these papers, the following paper was also composed during the project, however, Paper B encompass all of its main contributions and therefore it is not included in the thesis.

> J. Hansen, J. Østergaard, J. Kudahl, J. Madsen. "On the Construction of Jointly Superregular Lower Triangular Toeplitz Matrices". Published in in the proceedings of the *IEEE International Symposium on Information Theory*, pp. 1–5, July 2016.

# 4 Contributions

The main contributions of this thesis are (paper in parentheses):

1. Explicit non-asymptotic constructions of superregular lower triangular Toeplitz matrices over binary extension fields. (B)

2. Theoretical analysis, simulations, practical experiments of the symbol loss probability and symbol delay for block codes using superregular lower triangular Toeplitz matrices. (B)

3. Explicit constructions of convolutional codes with optimum distance profile with any code rate and degree over binary extension fields. (C)

4. Theoretical analysis, and simulations of the symbol loss probability and symbol delay for convolutional codes with optimum distance profile. (C)

5. A performance evaluation of random based network codes in low delay streaming, and the upper bound on the maximum code length of the code words produced by the intermediate recoder, with finite memory, using a rate $1/2$ convolutional network code, where both encoder and recoder use the same constraint length in a network with source, recoder, and sink. (D)

6. A performance evaluation of the sequential use of random based block codes and convolutional network codes in low delay streaming, and the upper bound on the maximum code length of the code words produced by the intermediate recoder, with finite memory, using a rate $1/2$ convolutional network code with constraint length $v_2$, where the encoder use a block or convolutional code with constraint length $v_1$ in a network with source, recoder, and sink. (E)

7. A flexible and extendable high performance software library for linear erasure correcting block and convolutional codes for both desktop computers and embedded devices, and a performance analysis of said library. (A)

8. Performance analysis of different software implementations of linear erasure correcting convolutional network codes. (A)

9. An investigation and discussion of some of the issues of using Wi-Fi for low delay media streaming. (A)

Paper A covers three of the main contributions. The paper present a new high performance software library for encoding, recoding and decoding of linear erasure correcting codes. The library is shown to be highly extendable and configurable, which makes it very versatile. The library is evaluated with

respect to throughput, for a set of different software implementations of linear erasure correcting codes. The set of software implementations covers both block codes and convolutional codes. For the case of block codes we compare the performance to that of a well known software library for linear block codes. The result of this comparison is that our implementation can provide up to 5.5x in throughput performance. Paper A then investigates several issues that have been discovered with respect to using Wi-Fi as the means for low delay media streaming. Finally, the paper discusses two channel models, namely the Gilbert-Elliott model and the extended Gilbert model, and closes with a presentation of a new channel model by extending the latter.

Paper B deals with the two first main contributions, by first presenting explicit non-asymptotic constructions of superregular lower triangular Toeplitz matrices and then going through a theoretical analysis of the performance of the block codes constructed from the superregular lower triangular Toeplitz matrices. The paper also includes a thorough set of simulations of the performance in a set of different scenarios. Some of these scenarios are based on a set of erasure patterns, which we obtained from a practical implementation.

In Paper C we present a new method for constructing convolutional codes of any rate with optimum distance profile. As part of this, the paper presents a theoretical analysis and a set of simulations measuring the symbol loss probability and symbol delay of the constructed convolutional codes. We also compare the performance of these codes with the performance of random based convolutional codes. Finally, an investigation of the effect of having finite decoder memory as oppose to infinite decoder memory is included.

Papers D and E investigate the loss performance, when the sink has strict delay requirements, for a number of erasure correcting network codes in a three node network with source, recoder, and sink. One of the results of Paper D is that, for low delay streaming with intermediate recoding, two concatenated triangular block codes are generally superior to two concatenated convolutional codes. Paper D also presents the upper bound on the maximum code length for the code words produced by a recoder using a rate $1/2$ convolutional network code with constraint length $v$ and memory factor $m$, when the encoder uses a convolutional code with the same parameters. In Paper E, it is shown that in some cases, the sequential use of triangular block codes and convolutional codes yields the lowest number of losses when employing strict delay constraints. Paper E presents the upper bound on the maximum code length for the code words produced by a recoder using a rate $1/2$ convolutional network code with constraint length $v_2$ and memory factor $m$, when the encoder use any block or convolutional code with constraint length $v_1$. Furthermore, Paper E uses simulation results to discuss the optimal parameters for the set of erasure correcting network codes given some channel and some maximum allowable symbol delay.

# Background and State-of-the-art

In this chapter we introduce the main topics of this PhD thesis and provide the reader with the background, state-of-the-art, and motivation for the work carried out.

## 1 Audio Formats

The Compact Disc Standard defines the audio format as 2 channels with a bit depth of 16 bits per sample and a sample rate of 44.1 kHz [1]; this format is known as CD quality. Although there is not a strict definition of high resolution audio, however, it may generally be defined as the formats capable of rendering beyond CD quality [2]. That is, formats with more than 16 bits per sample and/or a sample rate of more than 44.1 kHz would qualify as high resolution audio. On the other hand, the Blu-ray Disc format[1] allows for 6 channels of 24 bits per sample and a sample rate of 192 kHz or 8 channels of 24 bits per sample and a sample rate of 96 kHz. This is comparable to the requirements of HDMI (High-Definition Multimedia Interface) 2.0[2] which is 8 audio channels with a bit depth of 24 bits per sample and a sample rate of 192 kHz. For HDMI this amounts to a throughput of 36.864 $^{\text{Mb}}/_{\text{s}}$.

## 2 Low Delay Wireless Streaming of High Resolution Audio

A plethora of wireless enabled loudspeakers are available on the market. Many of these are based on some version of the Bluetooth specifications. However, the latest version, Bluetooth 5, only support transmissions with a data rate

[1] Audio Visual Application Format Specifications for BD-ROM Version 2.4 (www.blu-raydisc.com)

[2] See www.hdmi.org

of 2 Mb/s or optionally 3 Mb/s [3], which is more than an order of magnitude less than the throughput of HDMI. For this reason we do not consider Bluetooth 5 (and former versions) a viable means of achieving wireless streaming of high resolution audio with low delay.

An alternative wireless technology could be Wi-Fi, i.e., IEEE 802.11 and its amendments. This wireless technology has become the de facto standard for wireless internet for both domestic and corporate use. Amendment 802.11aa addresses some of the issues with respect to low delay streaming of multimedia to several receivers [3, 4]. One of the included mechanisms for increasing the reliability of multicast is *GCR Unsolicited Retry*. This allows the source to transmit packets to multiple receivers and then retransmit the packets several times without using acknowledgements. This method does not guarantee that every receiver has at least one successful packet reception, but it does ensure that at least some retransmissions are not delayed by automatic repeat requests. However, it does put a large amount of load on the shared medium, which may be unnecessary.

Another mechanism for reliable multicast in 802.11aa is *GCR Block Ack*. This allows the source to transmit a number of packets together as a single block, after which all receivers must acknowledge the block. In perfect conditions, the block may be then delivered with a low delay and high bandwidth. However, if packet losses are uncorrelated between receivers, a complete retransmission may be required. Furthermore, the multicast originator must initiate the Block Ack agreements with each receiver, which in turn may delay the transmission when the number of multicast recipients is large. The performance of these two mechanisms is highly dependent on the scenario [5].

## 2.1 Definition of Delay

According to [6, 7], in order to ensure an acceptable user experience, the timing between a video and the corresponding sound should not exceed 25 ms. This is not a strict latency limit for the audio distribution chain per se, but for live events where buffering is not possible, this limit may become strict. This means that in cases where the video part cannot be delayed, this limit restricts the allowable latency of the audio distribution chain. Given that such strict requirements exist, we have investigated the delay associated with forward erasure correcting codes.

The term delay has a plethora of different definitions and meanings. In this work, we have used two different definitions, in separate contexts. The following two sections describe our definitions, and present a motivation for both of them.

[3] See www.bluetooth.com

### Packets as Delay

In Papers B and C we define delay as the number of packets transmitted from the source, and possibly relays, after the source obtained symbol $x$, before the $x$'th symbol is decoded at the sink. Put in other words, we count the packets that are transmitted between the source obtains a specific symbol and until the sink decode said symbol. For the symbol in question, the number of packets is the experienced delay. The delay of undecodable symbols is set to zero. Alternatively, we could have set the delay of undecodable symbols to infinity, but since we are interested in the average delay, that would significantly influence the average delay. This definition of delay encapsulates the fact that packet transmissions represent a cost for the network. This cost may result in more than just a reduction in the available bandwidth – it may also increase the end-to-end latency. Furthermore, this also puts an emphasis on the rate of erasure correcting codes, e.g., the use of rate $1/2$ codes versus rate $1/3$ codes.

### Symbols as Delay

In Papers D and E we use another definition of delay, namely: We consider the number of symbols obtained by the source between the source obtains a specific symbol and the sink decodes said symbol. That is, when the sink decodes the $n$'th symbol and the source has obtained the $m$'th symbol, then the delay experienced by the $n$'th symbol is $m - n$, note that $m \geq n$. This definition of delay gives a one-to-one mapping between delay and the rate at which symbols are obtained by the source, denoted the input symbol rate. For many practical applications, such as audio streaming, the input symbol rate is fixed. This allows the designer of an erasure correcting code to utilize the input symbol rate and any latency constraints that may be set by the application to optimize some performance criteria, such as the loss probability.

## 3 Erasure Correcting Codes

In the paper "A mathematical theory of communication" [8] Claude Elwood Shannon presented an application of probability theory in order to study and analyse communication systems. The paper was later renamed to "The mathematical theory of communication" [9], a subtle but significant change, which was made after realizing the generality of the work.

The subject of this thesis revolves around erasure correcting codes, where erasures occur on a packet level. That is, we do not consider erasures of individual bits, but only erasures of entire packets. This type of code is fundamentally different from error correcting codes that correct errors (or erasure) on a bit or symbol level [10]. Erasure correcting codes are used to compensate for the

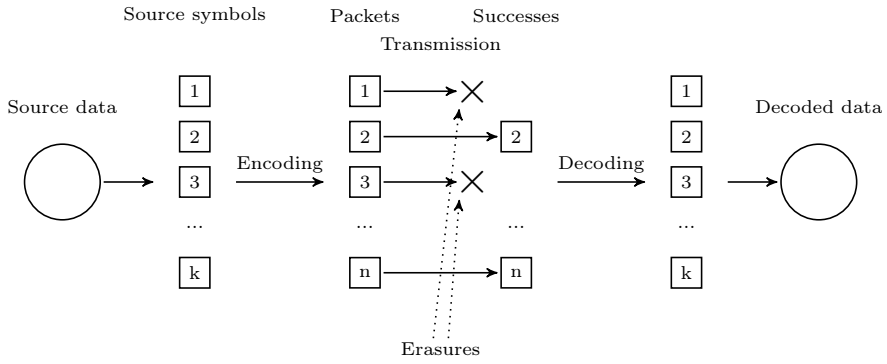



**Fig. 1:** Illustration of the encoding, transmission and decoding process.

packet erasures that occurs when an bit error correcting code is unable to recover from too many errors in a packet [11]. A common design of wireless communications systems [12, 13] is to use a bit error correcting code on the physical layer of the OSI model [14] and then use some form of checksum to verify the content of an entire packet on the data link layer [14]. Furthermore, a checksum may also employed on the transport layer [14]; this is the case for the two most commonly used transport protocols TCP [15, 16] and UDP [17] (optional with IPv4 [18, 19] and mandatory with IPv6 [20, 21]).

Fig. 1 shows the route from source data to the decoded data. The source data is segmented into $k$ symbols, these symbols are then encoded into $n$ packets. The packets are transmitted over an erasure channel to the receiver. The receiver then decodes the successfully received packets, and if enough packets has been received, recover all the data. In the event that the receiver does not receive enough packets, then some of the symbols may be recovered, depending the code and its structure. Since this is in fact a quite general representation of erasure correcting codes in communications systems, Fig. 1 has a cunning resemblance to that of Fig. 1 in [8], even though the present figure resolves around packet erasures and the figure in [8] is of a more general nature.

One of the advantages of erasure correcting codes is that enables a trade-off between loss probability, delay, and bandwidth [22]. By adding more redundancy, the available bandwidth is reduced but this may also reduce the loss probability, possibly without significantly affecting the delay. On the other hand, if reliable communication is of the utmost importance, then we may reduce the bandwidth and increase the delay, to ensure that the loss probability is minimized.

### 3.1 Linear Codes

In this work we only consider linear erasure correcting codes [23]. Moreover, linear means that the encoding (and decoding) step in Fig. 1 is a linear transformation. Furthermore, our codes operate over the finite field GF($2^p$), binary extension fields.

A code is termed systematic if the output of the encoder includes all the input symbols uncoded. Systematic codes are widely used [24–27], in part because it simplifies the encoding process and minimizes the encoding latency. Furthermore, it also simplifies the decoding process, since systematic packets require no decoding and can be forwarded immediately.

### 3.2 Random Linear Codes

In [28] T. Ho et al. introduced the use of random coding coefficients. One immediate benefit of using random coding coefficients is that it removes the need for designing encoding matrices with special properties. Instead, it is only the structure of the encoding matrix that needs to be designed [29–33]. Another benefit of random based codes is that optimal decoding capabilities can be shown through asymptotic analysis, when the size of the finite field and the dimensions of the code are sufficiently large [34]. Even for small code sizes and the finite fields GF($2^p$) with $p > 1$ random based dense block codes show near optimal decoding capabilities [35], on average.

## 4 Block Codes

When data is encoded and decoded in consecutive groups, it is termed a block code [36]. We define the input and output to and from the encoding process in terms of blocks, which may be described by matrices. We now illustrate the encoding procedure for block codes. The matrix $S$ contains the $k$ source symbols, and the matrix $C$ contains the $n$ packets. The $j$'th packet is generated using the $j$'th row of the encoding matrix $A$, like so:

$$C_j = \sum_{i=1}^{k} a_{j,i} S_i,$$

where $S_i$ is the $i$'th row of the source data matrix, and $C_j$ is the $j$'th row of the coded data matrix. That is, $C$ may be constructed as the product of $A$ and $S$:

$$C = AS.$$

Decoding is carried out by means of Gaussian elimination on the rows of $A$ that correspond to the received packets. Decoding requires $k(1 + \epsilon)$ packets, where a code is considered optimal if and only if $\epsilon = 0$. For codes where $\epsilon \to 0$

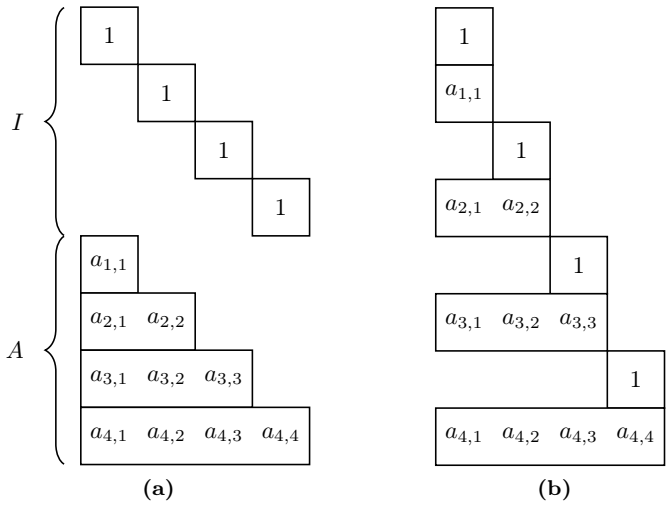



**Fig. 2:** (a) Shows a tall matrix consisting of two lower triangular matrices. (b) shows the rows of (a) rearranged.

is may be considered a near optimal code, this is the case for some random based codes [35].

## 4.1 Dense Codes

A linear block code is called maximum distance separable (MDS) if it attains the Singleton bound [37], and for linear erasure correcting block codes this mean that $\epsilon = 0$. That is, decoding any $k$ packets of the $n$ generated by the encoder allows for the decoding of all $k$ symbols. Dense codes are generally not suited for low delay communications, given that the entire set of input symbols must be present before encoding of any packet can take place. Thus, the encoder must wait for all symbols to be obtained, before it can encode and transmit any packets.

## 4.2 Lower Triangular Codes

Triangular codes are essentially block codes where every entry above the "diagonal" in the encoding matrix are zero; note that the definition of diagonal is in this context somewhat vague. We now show an example of an encoding matrix for a rate $1/2$ systematic block code with lower triangular structure. Fig. 2 shows two perturbations of an encoding matrix. Fig. 2a shows a tall matrix, which consist of two lower triangular matrices, i.e., $I$ and $A$, where $I$ is the identity matrix. When the encoding matrix is on this form it is easy to identify the two individual lower triangular matrices. In Fig. 2b the two matrices $I$

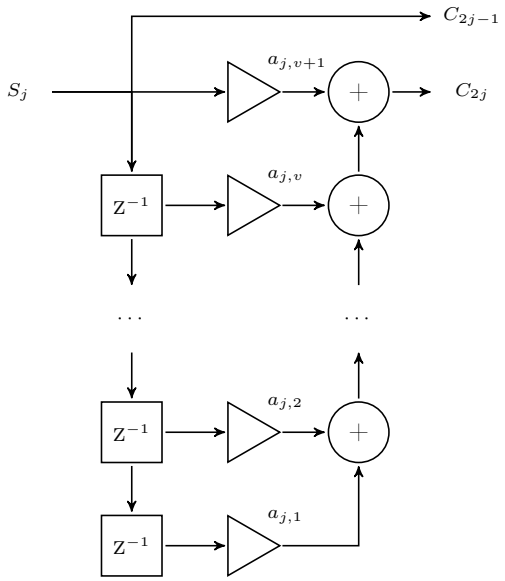



**Fig. 3:** Illustration of the encoding process for a systematic convolutional code with rate $1/2$ and memory $v$.

and $A$ have been merged. This form also show the order in which the packets are in, when transmitted over an erasure channel.

Lower triangular codes inherit a low delay property from the lower triangular structure of the individual lower triangular matrices, which the encoding matrix consist of. That is, the encoding delay is minimized, since the source can code and transmit packets immediately after obtaining a new source symbol. This structure also allows the sink to decode the received packets and extract the symbols on-the-fly, if enough packets have been received, and thereby reducing the decoding delay. The, potentially, large number of coefficients set to zero also increases the coding throughput [38].

# 5 Convolutional Codes

When the erasure correcting code is not segmented in blocks but used continuously, it is termed a convolutional codes [39]. Fig. 3 shows the encoding of two packets, when the source obtains the symbols $S_j$. That is, the figure shows encoding of packets using a systematic convolutional code, with rate $1/2$ and memory $v$, since the diagram in the figure contains $v$ delay elements. Fig. 4 shows an example of a continuous encoding matrix for a systematic convolutional code with rate $1/2$ and memory $v = 3$. The receiver may decode using Gaussian elimination, as previously stated about linear codes in general.

For random based convolutional codes there is little to no correlation between the matrix elements, $a_{j,i}, \forall j, i$. On the other hand, for static codes we may define $a_{j,i} = a_{u,i}, \forall j, u, i$. Using this definition we can truncate the continuous encoding matrix in Fig. 4; this is shown in Fig. 5a. By rearranging the rows we obtain the matrix in Fig. 5b, and we observe that the bottom part of the matrix is a lower triangular Toeplitz matrix. Since the matrix is Toeplitz me can simplify the matrix elements using $a_{j,i} = b_i, \forall j$, resulting in the matrix in Fig. 5c.

Now, let $\mathcal{C}$ be an $(n, k, \delta)$ systematic convolutional code with basic minimal generator matrix $G$. We then define $\delta$, the degree of $\mathcal{C}$, as the sum of the row degrees of $G$ [40]. We show in Paper C how the matrix in Fig. 5c may be used to form $G$.

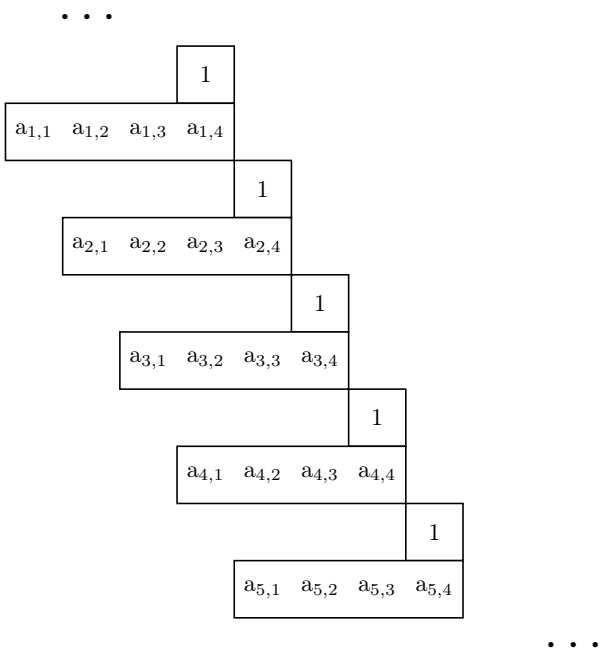



**Fig. 4:** Continuous encoding matrix for a rate 1/2 systematic convolutional code with memory $v = 3$.

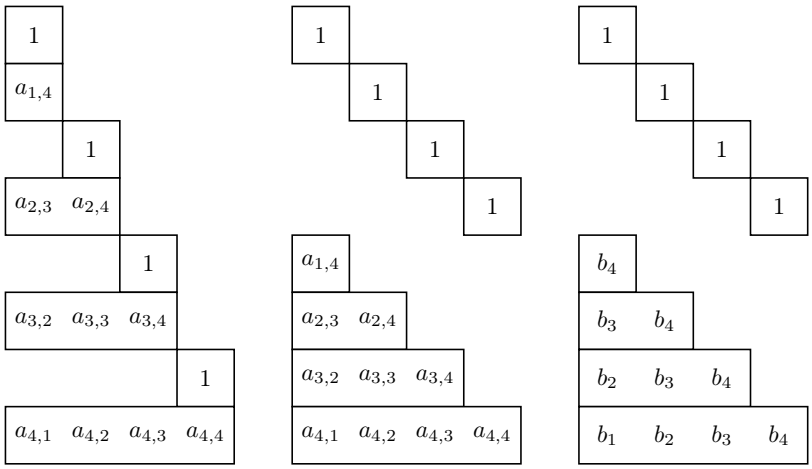



**Fig. 5:** Variations of the truncated continuous encoding matrix, in Fig. 4.

# 6 Superregular Matrices

Superregular matrices are a class of matrices which have been studied extensively [40–48]. The overall definition of superregularity is as follows; a matrix is superregular if and only if every proper submatrix is non-singular. The definition of a proper submatrix depends on the structure of the matrix. The following sections covers dense matrices and matrices with a lower triangular structure.

Superregular matrices are especially interesting for communication systems since their structure enable forward erasure correcting codes to have optimal decoding performance [49]. In [38, Definition 1] the authors defined optimal decoding capability. The definition fundamentally states that a code has optimal decoding capability if and only if there does not exist a code with the same structure of the encoding matrix that can recover more symbols over any erasure channel.

## 6.1 Dense Superregular Matrices

The authors of [49] defined a dense matrix to be superregular if and only if every square submatrix is non-singular. A dense superregular matrix can be constructed using a tall Vandermonde matrix [50, 51]. Let $n, m, p \in \mathbb{N}$ and let $n + m < 2^p$, then a $\mathrm{GF}(2^p)^{m \times n}$ superregular dense matrix may be constructed using the following matrices $A \in \mathrm{GF}(2^p)^{n+m \times n}$, $B \in \mathrm{GF}(2^p)^{n \times n}$

and $C, D \in \mathrm{GF}(2^p)^{m \times n}$

$$A = \begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \dots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \dots & \alpha_2^{n-1} \\ 1 & \alpha_3 & \alpha_3^2 & \dots & \alpha_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_{n+m} & \alpha_{n+m}^2 & \dots & \alpha_{n+m}^{n-1} \end{bmatrix} = \begin{bmatrix} B \\ C \end{bmatrix},$$

$$AB^{-1} = \begin{bmatrix} I \\ D \end{bmatrix}.$$

It follows that $I$ is the $n \times n$ identity matrix and that $D$ is a $m \times n$ superregular matrix if $\alpha_i \neq \alpha_j, i \neq j$ and $\alpha_i \neq 0, i \in \{1, \dots, n+m\}$.

Dense superregular matrices are appealing for erasure correcting codes, since they may be used to form systematic block codes that are MDS [49]. That is, the matrix $AB^{-1}$ may be used to form such a code, since the upper matrix is the identity matrix and the lower part is a dense superregular matrix.

## 6.2 Lower Triangular Superregular Matrices

The authors of [40] defined a lower triangular matrix to be superregular if and only if every proper submatrix is non-singular. The following definition of a proper submatrix follows from [38, 40]. Let $A$ be a $k \times k$ lower triangular matrix. Let $A' = A^{j_1,\dots,j_r}_{h_1,\dots,h_r}$ be an $r \times r$ submatrix of $A$, where $A'$ is constructed using the rows and columns of $A$ with indices $j_1, \dots, j_r$ and $h_1, \dots, h_r$, respectively. Then, $A'$ is a proper submatrix of $A$ if and only if $1 \leq j_1 < j_2 < \dots < j_r \leq k$, $1 \leq h_1 < h_2 < \dots < h_r \leq k$ and $j_t \geq h_t, \forall t$.

In [40], the authors presented a few superregular lower triangular Toeplitz matrices. However, no insights were given as to how they were constructed. In [52], a construction for superregular (totally positive) matrices was provided for real and complex fields. This construction can easily be extended to very large prime fields, although these are impractical. In [43, Example 6] the authors presented a similar method to construct superregular lower triangular Toeplitz matrices of any size over a sufficiently large prime field. The method is as follows. Let $k \in \mathbb{N}$, and $X$ be a $(k+1) \times (k+1)$ matrices with the following structure:

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

Then $X^k$ is a superregular matrix over a sufficiently large prime field. Unfortunately, no bounds on the field size was provided.

There are two major of drawbacks of this method. First, the use of large prime fields is not feasible in practical implementations. Second, this method constructs a unique matrix of size $k \times k$. That is, the method cannot produce two or more different matrices with same $k$. Also note the fact that CPUs use a binary representation of integers, and that binary extension field arithmetic does not require modulo operations. For this reason, it is generally considered straight forward to implement $\mathrm{GF}(2^p)$ arithmetic in software [53], whereas large prime fields are significantly more complex to implement and use in practical applications.

### 6.3 Using Superregular Matrices

Superregular matrices have been used to form both block codes [49] and convolutional codes [40, 42, 44–46, 48]. In [40] the authors introduced a new class of MDS convolutional codes whose column distances attain the generalized Singleton bound at the earliest possible instant, these codes are termed strongly-MDS convolutional codes. In [44] R. Hutchinson et al. show how superregular matrices may be used to construct MDP convolutional codes, and present an upper bound on the minimum field size in order for a superregular matrix of a given size to exist over that field. The authors of [42] introduce the metric denoted column sum rank, which parallels the column Hamming distance when using a network with link failures. Furthermore, the authors also show a class of superregular matrices that preserve the superregular property after multiplication with non-singular block diagonal matrices. P. Almeida et al. showed in [45] new class of matrices that are superregular over a sufficiently large finite field. These matrices are used to construct MDP convolutional codes. In [46] the authors introduce a natural bound on the distance of a 2D convolutional code of rate k/n and degree $\delta$, which generalizes the Singleton bound for block codes and the generalized Singleton bound for 1D convolutional codes. In [48] the authors presented constructions of convolutional codes that attain the maximum possible distance for some fixed parameters of the code, namely: the rate and the Forney indices.

## 7 Network coding

Network coding is the concept of combining packets in the network, and not only at the end nodes. The concept of network coding was introduced by the authors of [54]. In general, this is a method that enables either a higher throughput [55] or resilience against packet loss [56], known as inter-session and intra-session network coding respectively. Using inter-session network coding, packets are only combined with packets from other flows in order to increase throughput [57, 58]. The concept of inter-session network coding is not cov-

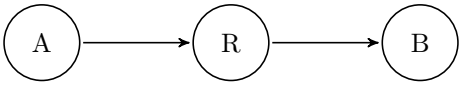



**Fig. 6:** A multi-hop network with three nodes. A transmit packets to R, which then recodes the received packets and transmit the recoded packets to B.

ered in this thesis. With intra-session network coding, packets are only coded together with packets from the same flow in order to increase reliability, by adding redundancy.

## 7.1 Intra-Session Network Coding

Fig. 6 shows a network where an intermediate node, R, performs recoding. This technique allows for two different code rates on the two links. That is, the redundancy may be added to exactly the links where it is needed. Alternatively, the redundancy would have to be carried on all links, even if some of these links are error free.

A lot of research efforts have been put into the area of intra-session network coding [59–61]. In [59] the authors presented an opportunistic routing scheme using intra-session network coding for wireless mesh networks, where the inherent broadcast nature of wireless networks are exploited. In [60] T. Ho et al. considered the problem of multiple multicast sessions with intra-session network coding within dynamic networks. This work leads to the establishment of a capacity region of input rates that can be stably supported at the network-layer. The authors of [61] presented adaptive rate control algorithms for the networks with or without given coding subgraphs.

# 8 Embedded Platforms

Embedded platforms have historicity been limited in some form, either in computational power, memory or simply power (e.g., battery powered) or a combination of these limitations. While embedded platforms still have limitations, compared to desktop computers, they may contain the computational power to encode and decode complex erasure correcting codes [27]. From an engineering perspective, it is of the utmost importance to optimise the software (with respect to both energy and/or throughput) which these platforms execute, given that these limitations exist. Furthermore, the erasure correcting codes may also be designed to improve energy efficiency [38], by reducing the computational resources required.

### Power Consumption in Receiving and Idle State

In [62] it was found that for some wireless network interface cards (WNIC), the power consumption in receiving state can be an order of magnitude larger than when in idle state, but still connected. Thus, there can be a large incitement to use power only for receiving innovative packets, i.e., packets that are linearly independent of previously received packets. This may not possible if the code words are generated using random coefficients. However, by using a static code low-power embedded devices are able to calculate in advance whether the next packet is innovative or not. If the next packet is not innovative the device should not receive it. This method allows low-power devices to save power, by not activating the receiver circuit to receive non-innovative packets.

# Conclusions

The work documented in this thesis covers the following three research areas:

1. An information theoretical result regarding the construction of (jointly) superregular matrices and a theoretical analysis of the performance of linear erasure correcting codes.

2. An extensive set of simulations of different networks with and without recoding at intermediate nodes. These simulations have led to several findings regarding the performance of erasure correcting codes and the applicability of different codes to different scenarios.

3. A comprehensive set of practical tests, conducted on a robust experimental Wi-Fi testbed.

Finally, these three areas have been related to each other in order to discuss the nexus between theory and practice.

Paper A showed several new results on software implementations of linear erasure correcting codes. The paper also presented an experimental Wi-Fi testbed that enabled repeatable and consistent tests. The testbed seeks to eliminate external noise sources by using wires instead of antennas for the Wi-Fi connections. Furthermore, said paper also presented new results on the Extended Gilbert Model; this channel model was then extended to allow for bursts of arbitrary length and to reduce the parameter complexity.

Several results on the construction and performance of linear erasure correcting codes have been presented in this thesis. A set of explicit constructions of linear erasure correcting block codes is covered in Paper B. These block codes are based on superregular lower triangular Toeplitz matrices, which the paper also provided explicit constructions for. Moreover, an algorithm for constructing a $k \times k$ superregular lower triangular Toeplitz matrix over any binary extension field is also presented in said paper. These explicit constructions and algorithm for constructing superregular lower triangular Toeplitz matrix matrices outperform the state-of-the-art with respect to the required size of the finite field. Furthermore, these methods allow for the construction of said matrices over small binary extension fields, as opposed to very large prime fields. This

is then followed up by Paper C which provides explicit constructions of linear erasure correcting convolutional codes with optimum distance profile over any binary extension field.

For a network with recoding, Paper D compared the symbol loss performance, given strict delay constraints for three selected code structures. The code structures are two block codes, namely dense and lower triangular codes, and the third structure is convolutional codes. It was shown that using a lower triangular structure for both encoding and decoding is, in general, superior to the other two structures, regardless of the delay constraints and the erasure probability of the channel. In Paper E we extended this work to include a mixture of block and convolutional codes. That is, we allowed for the recoder to use a convolutional network code when the encoder used a block code. Generally, the use of a lower triangular code for encoding and a convolutional code for recoding proved to be superior to the remaining schemes, regardless of the delay constraints and the erasure probability of the channel. However, this may not be attractive from a practical perspective, given that the algorithmic complexity is significantly higher when combining the coding schemes. Furthermore, as it was shown in Paper A the coding throughput of software implementations of block codes may be considerably higher than the throughput for convolutional codes, which partly comes from the lower algorithmic complexity of block codes.

Although the concept of low delay erasure correcting codes have been subject to research for a considerable number of decades, we strongly believe that much can be learnt within this field of research. For example, the authors of Paper C have been unable to extend the analysis of the exact symbol loss and delay of convolutional codes with finite decoder memory to the asymptotic case. Such a result would enable the calculation of the average symbol delay and loss probability for convolutional codes with finite decoder memory.

# References


[1] IEC 60908:1999, "Audio recording — Compact disc digital audio system," February 1999.

[2] J. D. Reiss, "A meta-analysis of high resolution audio perceptual evaluation," *Journal of the Audio Engineering Society*, vol. 64, no. 6, pp. 364–379, June 2016.

[3] K. Kosek-Szott, M. Natkaniec, S. Szott, A. Krasilov, A. Lyakhov, A. Safonov, and I. Tinnirello, "What's new for QoS in IEEE 802.11?" *IEEE Network*, vol. 27, no. 6, pp. 95–104, November 2013.

[4] E. Charfi, L. Chaari, and L. Kamoun, "PHY/MAC Enhancements and QoS Mechanisms for Very High Throughput WLANs: A Survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 1714–1735, April 2013.

[5] A. Banchs, A. de la Oliva, L. Eznarriaga, D. R. Kowalski, and P. Serrano, "Performance Analysis and Algorithm Selection for Reliable Multicast in IEEE 802.11aa Wireless LAN," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 8, pp. 3875–3891, October 2014.

[6] ITU-R BT.1359-1, "Relative Timing of Sound and Vision for Broadcasting," 1998.

[7] C. Howson, E. Gautier, P. Gilberton, A. Laurent, and Y. Legallais, "Second screen TV synchronization," *IEEE International Conference on Consumer Electronics*, pp. 361–365, September 2011.

[8] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[9] C. Shannon and W. Weaver, *The Mathematical Theory of Communication*, ser. Illini books. University of Illinois Press, 1963.

[10] R. E. Blahut, *Theory and practice of error control codes*. Addison-Wesley Publishing Company, 1983, vol. 126.

[11] L. Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 2, pp. 24–36, April 1997.

[12] E. Perahia and R. Stacey, *Next Generation Wireless LANs: 802.11n and 802.11ac*, 2nd ed. Cambridge University Press, 2013.

[13] L. Lampe, M. Jain, and R. Schober, "Improved decoding for bluetooth systems," *IEEE Transactions on Communications*, vol. 53, no. 1, pp. 1–4, January 2005.

[14] ISO/IEC 7498-1:1994, "Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model," November 1994.

[15] V. Cerf, Y. Dalal, and C. Sunshine, "Specification of Internet Transmission Control Program," RFC 675 (Historic), RFC Editor, pp. 1–70, December 1974.

[16] A. Zimmermann, W. Eddy, and L. Eggert, "Moving Outdated TCP Extensions and TCP-Related Documents to Historic or Informational Status," RFC 7805 (Informational), RFC Editor, pp. 1–8, April 2016.

[17] J. Postel, "User Datagram Protocol," RFC 768 (Internet Standard), RFC Editor, pp. 1–3, August 1980.

[18] J. Postel, "DoD standard Internet Protocol," RFC 760, RFC Editor, pp. 1–46, January 1980.

[19] J. Touch, "Updated Specification of the IPv4 ID Field," RFC 6864 (Proposed Standard), RFC Editor, pp. 1–19, February 2013.

[20] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 1883 (Proposed Standard), RFC Editor, pp. 1–37, December 1995.

[21] F. Gont, V. Manral, and R. Bonica, "Implications of Oversized IPv6 Header Chains," RFC 7112 (Proposed Standard), RFC Editor, pp. 1–8, January 2014.

[22] M. Muntner and J. Wolf, "Predicted performance of error-control techniques over real channels," *IEEE Transactions on Information Theory*, vol. 14, no. 5, pp. 640–650, September 1968.

[23] V. Pless, *Introduction to the Theory of Error-correcting Codes*, ser. A Wiley-Interscience publication. Wiley-Interscience, 1989.

[24] D. E. Lucani, M. Médard, and M. Stojanovic, "Systematic network coding for time-division duplexing," *IEEE International Symposium on Information Theory*, pp. 2403–2407, June 2010.

[25] C. Heegard, J. Little, and K. Saints, "Systematic encoding via grobner bases for a class of algebraic-geometric goppa codes," *IEEE Transactions on Information Theory*, vol. 41, no. 6, pp. 1752–1761, November 1995.

[26] J. Macwilliams, "Permutation decoding of systematic codes," *The Bell System Technical Journal*, vol. 43, no. 1, pp. 485–505, January 1964.

[27] J. Heide, M. V. Pedersen, F. H. P. Fitzek, and T. Larsen, "Network coding for mobile devices — systematic binary random rateless codes," *IEEE International Conference on Communications Workshops*, pp. 1–6, June 2009.

[28] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," *IEEE International Symposium on Information Theory*, June 2003.

[29] S. Feizi, D. E. Lucani, C. W. Sørensen, A. Makhdoumi, and M. Médard, "Tunable sparse network coding for multicast networks," *International Symposium on Network Coding*, pp. 1–6, June 2014.

[30] J. Heide, M. V. Pedersen, F. H. P. Fitzek, and M. Médard, "A perpetual code for network coding," *IEEE 79th Vehicular Technology Conference*, pp. 1–6, May 2014.

[31] D. E. Lucani, M. V. Pedersen, J. Heide, and F. H. P. Fitzek, "Coping with the upcoming heterogeneity in 5G communications and storage using Fulcrum network codes," *11th International Symposium on Wireless Communications Systems*, pp. 997–1001, August 2014.

[32] S. Wunderlich, F. Gabriel, S. Pandi, F. H. P. Fitzek, and M. Reisslein, "Caterpillar RLNC (CRLNC): A Practical Finite Sliding Window RLNC Approach," *IEEE Access*, vol. 5, pp. 20 183–20 197, 2017.

[33] P. Garrido, D. Gómez, J. Lanza, and R. Agüero, "Exploiting sparse coding: A sliding window enhancement of a random linear network coding scheme," *IEEE International Conference on Communications*, pp. 1–6, May 2016.

[34] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, October 2006.

[35] J. Heide, M. Pedersen, F. Fitzek, and M. Médard, "On Code Parameters and Coding Vector Representation for Practical RLNC," *IEEE International Conference on Communications*, pp. 1–5, June 2011.

[36] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, April 1950.

[37] F. MacWilliams and N. Sloane, *The Theory of Error-Correcting Codes*, ser. North-Holland mathematical library. North-Holland Publishing Company, December 1977.

[38] J. Hansen, J. Østergaard, J. Kudahl, and J. H. Madsen, "Superregular lower triangular toeplitz matrices for low delay wireless streaming," *IEEE Transactions on Communications*, vol. 65, no. 9, pp. 4027–4038, September 2017.

[39] P. Elias, "Coding for noisy channels," *IRE Convention Records Part 4*, pp. 37–46, 1955.

[40] H. Gluesing-Luerssen, J. Rosenthal, and R. Smarandache, "Strongly-MDS convolutional codes," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 584–598, February 2006.

[41] R. Mahmood, A. Badr, and A. Khisti, "Convolutional codes with maximum column sum rank for network streaming," *IEEE International Symposium on Information Theory*, pp. 2271–2275, June 2015.

[42] R. Mahmood, A. Badr, and A. Khisti, "Convolutional codes with maximum column sum rank for network streaming," *IEEE Transactions on Information Theory*, vol. 62, no. 6, pp. 3039–3052, June 2016.

[43] R. Smarandache, H. Gluesing-Luerssen, and J. Rosenthal, "Strongly MDS convolutional codes, a new class of codes with maximal decoding capability," *IEEE International Symposium on Information Theory*, January 2002.

[44] R. Hutchinson, R. Smarandache, and J. Trumpf, "On Superregular Matrices and MDP Convolutional Codes," *Linear Algebra and its Applications*, vol. 428, no. 11-12, pp. 2585–2596, 2008.

[45] P. Almeida, D. Napp, and R. Pinto, "A new class of superregular matrices and MDP convolutional codes," *Linear Algebra and its Applications*, vol. 439, no. 7, pp. 2145–2157, 2013.

[46] J. J. Climent, D. Napp, C. Perea, and R. Pinto, "Maximum Distance Separable 2D Convolutional Codes," *IEEE Transactions on Information Theory*, vol. 62, no. 2, pp. 669–680, February 2016.

[47] J. Hansen, J. Østergaard, J. Kudahl, and J. H. Madsen, "On the Construction of Jointly Superregular Lower Triangular Toeplitz Matrices," *IEEE International Symposium on Information Theory*, pp. 1–5, July 2016.

[48] P. Almeida, D. Napp, and R. Pinto, "Superregular matrices and applications to convolutional codes," *Linear Algebra and its Applications*, vol. 499, pp. 1–25, June 2016.

[49] R. Roth and A. Lempel, "On MDS codes via Cauchy matrices," *IEEE Transactions on Information Theory*, vol. 35, no. 6, pp. 1314–1319, November 1989.

[50] J. Lacan and J. Fimes, "Systematic MDS erasure codes based on Vandermonde matrices," *IEEE Communications Letters*, vol. 8, no. 9, pp. 570–572, September 2004.

[51] J. Lacan, V. Roca, J. Peltotalo, and S. Peltotalo, "Reed-Solomon Forward Error Correction (FEC) Schemes," RFC 5510 (Proposed Standard), RFC Editor, pp. 1–28, April 2009.

[52] M. Aissen, I. Schoenberg, and A. Whitney, "On the generating functions of totally positive sequences I," *Journal d'Analyse Mathématique*, vol. 2, no. 1, pp. 93–103, 1952.

[53] A. Paramanathan, P. Pahlevani, S. Thorsteinsson, M. Hundebøll, D. E. Lucani, and F. H. P. Fitzek, "Sharing the Pi: Testbed Description and Performance Evaluation of Network Coding on the Raspberry Pi," *IEEE 79th Vehicular Technology Conference*, 2014.

[54] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[55] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the Air: Practical wireless network coding," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 497–510, June 2008.

[56] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 169–180, August 2007.

[57] J. Krigslund, J. Hansen, M. Hundebøll, D. Lucani, and F. Fitzek, "CORE: COPE with MORE in Wireless Meshed Networks," *IEEE 77th Vehicular Technology Conference*, June 2013.

[58] H. Seferoglu, A. Markopoulou, and K. Ramakrishnan, "I2NC: Intra- and Inter-session Network Coding for Unicast Flows in Wireless Networks," *IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1035–1043, April 2011.

[59] B. Radunovic, C. Gkantsidis, P. Key, and P. Rodriguez, "Toward practical opportunistic routing with intra-session network coding for mesh networks," *IEEE/ACM Transactions on Networking*, vol. 18, no. 2, pp. 420–433, April 2010.

[60] T. Ho and H. Viswanathan, "Dynamic algorithms for multicast with intra-session network coding," *IEEE Transactions on Information Theory*, vol. 55, no. 2, pp. 797–815, February 2009.

[61] L. Chen, T. Ho, S. H. Low, M. Chiang, and J. C. Doyle, "Optimization based rate control for multicast with network coding," in *IEEE 26th International Conference on Computer Communications (INFOCOM)*, May 2007, pp. 1163–1171.

[62] S. Chiaravalloti, F. Idzikowski, Ł. Budzisz, and A. Wolisz, "Power Consumption of WLAN Network Elements," *Technische Universität Berlin*, Technical Report TKN-11-002, 2011.

# Part II

# Papers

# Paper A

## Network Coding on Embedded Platforms for Wireless Networks

Jonas Hansen, Jan Østergaard, Johnny Kudahl, and John H. Madsen

Technical report.

# Abstract

*In this paper we introduce a network coding software library and demonstrate the performance of said library on both x86 and ARM architectures. This library is capable of encoding, recoding, and decoding both linear block and convolutional erasure correcting codes. For the case of dense block codes, we compare the performance with a well known network coding software for linear block codes. We compare the performance of three different implementations of convolutional codes. We document benefits and drawbacks, with respect to real-time streaming, of three casting methods for Wi-Fi networks. We show an experimental Wi-Fi setup, which is used to model bursty erasure patterns. We introduce a new channel model for the erasure channel.*


# 1 Introduction

There are several unanswered research questions with respect to the implementation of erasure correcting codes in software. There is a plethora of different ways to implement erasure correcting codes, and a subset of these possible methods must be more efficient than others. In order to evaluate which implementation of block codes and convolutional codes performs best, we have implemented and measured the throughput performance of a set of implementations.

One of the key motivations for this measurement campaign is that the computational complexity of the erasure correcting codes should not exceed the capacity of modern embedded devices. Based on this we have implemented a high performance software library for erasure correcting codes on an ARM platform. Additionally, the software library can also run on Intel x86 processors, which we demonstrate firstly.

The throughput required for encoding and decoding multi channel high resolution audio, e.g, 8 audio channels with a bit depth of 24 bits per sample and a sample rate of 192 kHz (which is the minimum requirement for HDMI 2.0[1]), is[2]

$$8 \cdot 24 \cdot 192000 = 36.864 \ {}^{\mathrm{Mb}}/_{\mathrm{s}} = 4.608 \ {}^{\mathrm{MB}}/_{\mathrm{s}}.$$

That is, our software library should be able to encode and decode the data with a rate of 4.608 $^{\mathrm{MB}}/_{\mathrm{s}}$. This is of course an upper limit on the required throughput since in practise it will not be the same node doing both encoding and decoding, except for any intermediate nodes performing recoding.

In order to properly evaluate the performance of erasure correcting codes an accurate channel model is needed. To this end, we have built a Wi-Fi testbed

[1] See www.hdmi.org

[2] The units of the left most side of the equation have been left out for the sake of brevity.

capable of changing the actual Wi-Fi channel. The changes to the channel can both be in terms of SNR and fading, or a combination of the two. The erasure pattern of the Wi-Fi channel can then be recorded and modelled, this then enables the simulation of Wi-Fi channels with realistic erasure patterns.

Modelling these erasure patterns is non-trivial. Several channel models has been proposed, among them are the Gilbert-Elliott model [1, 2] and the extended Gilbert model [3]. The benefits and drawbacks of these models are discussed in Section 8. In order to accommodate the shortcomings of these models we propose an extension of the latter.

## 2 Source Code

In this section we present a software library is written in C++, and more specifically C++17 [4]. This software library is capable of encoding, recoding and decoding erasure correcting codes, either as block codes or as convolutional codes. The encoders, recoders, and decoders are collectively denoted coders. The software implementation of the coders uses the *parametrized inheritance* or *mixin* design method [5, 6]. This design method seeks to make the software modular, testable, and allow for easy code reuse.

Listing A.1 shows three examples of coders. The coders are named using type aliases to increase the readability of the source codes. The coders consist of at least four layers, namely: interface, algorithm, data storage, and a finite field. The interface defines which functions (operations) a coder supports. The one or more algorithm layers implement the data processing methods, e.g., encoding, decoding, or both. The storage layer defines how and where data is stored in memory. There are three types of data storage layers:

- **Shallow**: the coder does not provide any memory, but will use memory provided by the user.
- **Deep**: the coder allocates and deallocates memory upon construction and destruction, respectively.
- **Managed**: the coder implements the concept of a *free list* consisting of previously allocated (and released) memory.

An example of the code reuse capability of the mixin design is that both the deep and managed memory layers are implemented using the shallow memory layer. That is, for the deep memory layer the constructor allocates and sets the memory of the shallow memory layer, and the destructor frees the memory. The managed memory layer uses a similar method, but using an implementation that resembles a *free list* with support for different sizes. The finite field layer implements finite field arithmetic, which is used when processing the data.

**Listing A.1:** C++ example code for encoder, decoder, recoder types with managed memory.

```
using encoder =
    encoder_interface<      // The public functions of the encoder
    random_encoder<         // Encode using random coeffcients
    managed_storage<        // Memory manager
    gf2<8>>>>;              // Field: GF(2^8)

using decoder =
    decoder_interface<      // The public functions of the decoder
    bidirectional_decoder<  // Perform Gaussian elimination with both
                            // forward and backward substitution
    managed_storage<        // Memory manager
    gf2<8>>>>;              // Field: GF(2^8)

using recoder =
    encoder_interface<
    random_encoder<         // Encode using random coeffcients
    decoder>>;              // The decoder functions
```

**Listing A.2:** C++ example code for the *encoder_interface* class.

```
template <class super>
class encoder_interface : public super
{
public:
    encoder_interface(const uint32_t k, const uint32_t symbol_size)
     : super(k, symbol_size) {}

    template <class allocator>
    uint32_t encode(std::vector<std::byte, allocator>& payload)
    {
        assert(payload.size() >= super::payload_size());
        return super::encode(payload);
    }

    /* The remaing public function are not included here */
};
```

Note that the algorithm layer of the encoder in Listing A.1 uses random coefficients, and that this does not require any state information. In fact, this encoder will encode any symbol it knows with a random coefficient, and the set of known symbols may be subject to change between the encoding of two packets. That is, this encoder does not put any restrictions on the placement and number of non-zero coding coefficients in the coding matrix. If a Reed-Solomon code [7] is desired instead of random coefficients, then this can be changed by simply using the *rs_encoder* instead of the *random_encoder*. This is one of the benefits of using the mixin design.

Listing A.2 shows part of the implementation of the *encoder_interface* class. The class inherit from the class *super*, which is provided as a template parameter. The constructor only calls the constructor of super. The class defines the *encode* function, which only check the size of the provided payload and then

**Listing A.3:** C++ example code for triangular encoding and decoding.

```
encoder e(k, symbol_size);
decoder d(k, symbol_size);
std::vector<std::byte*> data = /* Fill with pointers to data */ ;
std::vector<std::byte> payload(e.payload_size());
while(not d.full_rank())
{
    if (auto r = e.rank(); r != k)
    {
        e.set_symbol(r, data[r]);
    }

    e.encode(payload);
    /* Packet loss can be emulated by randomly calling: continue */
    d.decode(payload);
}
```

forwards the call to *super::encode*. The coder is at instantiation required to know the number of symbols, $k$, in a block. Finally, the memory needed to store the symbol data is required to be known at instantiation of the coders, this is denoted the *symbol size*.

## 2.1 Encode and Decode Example

Listing A.3 present how to perform triangular encoding and decoding using the software library. The coders used in this example are defined in Listing A.1. The triangular structure is obtained by gradually providing symbols to the encoder. That is, until the encoder has all symbols, a symbol is added before encoding the next packet. The coding matrix does then get a triangular structure for the first $k$ packets, the following packets are then dense. The coding coefficients are chosen at random over the finite field GF($2^8$) [8]. The decoder performs Gaussian elimination on the received packets.

## 2.2 Encode, Recode, and Decode Example

Listing A.4 extends Listing A.3 by introducing a recoder between the encoder and decoder. In this example the encoder also produce a triangular coding structure. The recoder acts both as a decoder and a encoder, as it is made clear from its definition, in Listing A.1. By decoding the received packets the recoder ensures that all the stored packets are linear independent. By the same token, these linear independent packets are then encoded (recoded) and transmitted to the decoder.

**Listing A.4:** C++ example code for triangular encoding, recoding, and decoding.

```
encoder e(k, symbol_size);
decoder d(k, symbol_size);
recoder r(k, symbol_size);
std::vector<std::byte*> data = /* Fill with pointers to data */ ;
std::vector<std::byte> payload(e.payload_size());
while(not d.full_rank())
{
    if (auto r = e.rank(); r != k)
    {
        e.set_symbol(r, data[r]);
    }

    e.encode(payload);
    /* Packet loss can be emulated by randomly calling: continue */
    r.decode(payload);

    r.encode(payload);
    /* Packet loss can be emulated by randomly calling: continue */
    d.decode(payload);
}
```

# 3 Coding Throughput of Block Codes on x86

In this section we evaluate the throughput performance of the software library when using an Intel CPU. The details for the test node used for these performance evaluations are listed in Table A.1. Fig. A.1a and A.1b show the throughput with and without recoding, respectively. The encoding is triangular as shown in Listings A.3 and A.4. Table A.2 lists the four implementations of finite field arithmetic the software library supports. For multiplication and inversion both online and lookup table based arithmetic is supported. There are two implementations where both multiplications and inversions are based on lookup tables, however, for one of them multiplication is vectorized. Modern CPUs support vector arithmetic, that is, a single instruction can operate on $N \in \{128, 256, 512\}$ bytes simultaneously. These instructions are termed single instruction multiple data (SIMD) instructions, and the value of $N$ depend on the architecture of the CPU. For the *Vectorized* implementation multiplication is carried out using AVX2 instructions in the CPU, $N$ is 256 for AVX2.

From the figures it is clear that the primary element to ensure high performance is vectorization of finite field multiplication. For the case of the Vectorized implementation, comparing the performance with and without recoding shows the intuitive result that adding recoding halve the throughput. The gain in throughput of implementing inversion as a lookup table is not significant as for $k > 2$. Implementing multiplication via a lookup table provides about 10 % better performance than using an online implementation. For all tested values of $k$ the throughput of all four implementations is well above the

**Table A.1:** Details of the x86 node used in the experiments.

| Type | Name |
|---|---|
| CPU | Intel Core i3-4170 |
| OS | Ubuntu Linux 15.10 |
| Kernel | Linux 4.4.11 |

**Table A.2:** Description of the supported finite field implementations.

| Name | Multiplication | Inversion | Addition |
|---|---|---|---|
| Full online | Online | Online | Vectorized |
| Online product | Online | Lookup table | Vectorized |
| Lookup table | Lookup table | Lookup table | Vectorized |
| Vectorized | Vectorized | Lookup table | Vectorized |

required 4.608 MB/s. In fact, for $k = 2$ and no recoding, the throughput is more than three orders of magnitude higher for the vectorized implementation.

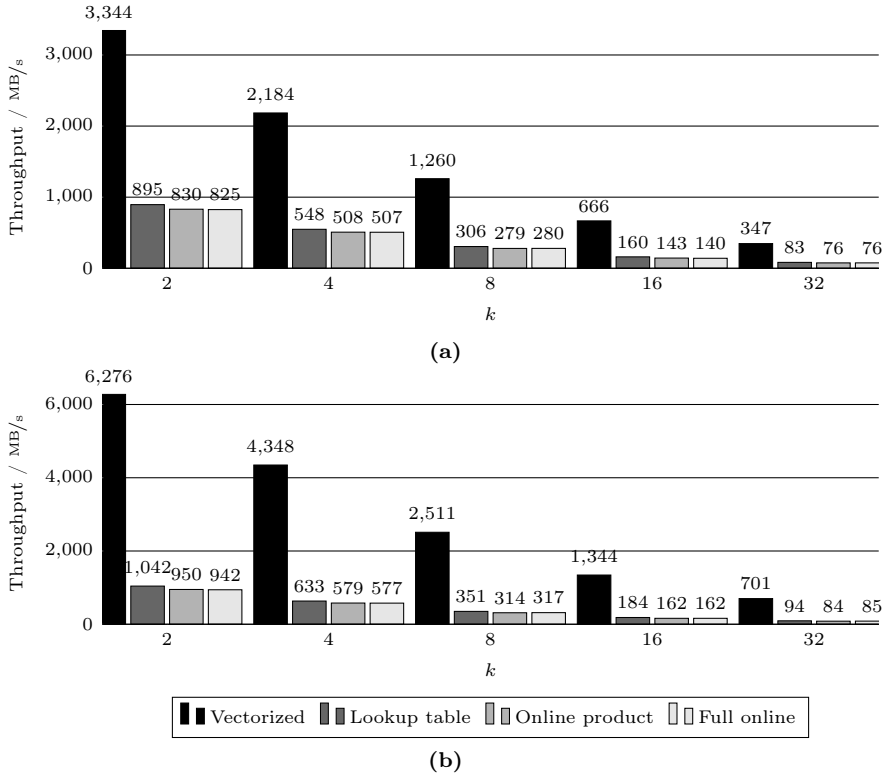



**Fig. A.1:** Coding throughput, on x86, when performing encoding, (for (a) also recoding), and decoding.

**Listing A.5:** C++ example code for dense encoding, recoding, and decoding.

```
tic(); // Start timing measurements
encoder e(k, symbol_size);
decoder d(k, symbol_size);
recoder r(k, symbol_size); // Recoding profile only
std::vector<std::byte> data_in(k * symbol_size);
std::vector<std::byte> payload(e.payload_size());
e.set_generation(data_in);
while(not d.full_rank())
{
    e.encode(payload);
    r.decode(payload); // Recoding profile only
    r.encode(payload); // Recoding profile only
    d.decode(payload);
}
toc(); // Stop timing measurements
```

## 3.1 Comparison to State-of-the-art

We now evaluate the performance of the software library when using dense codes and deep memory. The achieved performance is then compared to that of the Kodo version 20.0.0 [9]. Kodo is a software library that implements network coding for block codes. Listing A.5 shows how we measure the performance of our library, and Listing A.6 shows the same when using Kodo. The two listings both contain three lines marked with *Recoding profile only*, these lines are not present when measuring the performance without recoding. The key metric in this measurement process is the time spent between *tic* and *toc*, the throughput is calculated from these measurements. In order to produce reliable results, the code between *tic* and *toc* is repeated $10^5$ times.

Fig. A.2a and A.2b show the measured throughput for both software libraries with and without recoding, respectively. From the figures it is, again, clear that using recoding cuts the throughput in half. It is also clear that our software library is significantly faster than Kodo for $k < 8$, in fact for $k = 2$ and no recoding there is a gain of more than 5.5x. There is still a large gain of our software library for $k \geq 8$, on average this gain is 2.45x.

**Listing A.6:** C++ example code for network coding using Kodo.

```
kodo::rlnc::full_vector_encoder<fifi::binary8>::factory
    encoder_factory(k, symbol_size);
kodo::rlnc::full_vector_decoder<fifi::binary8>::factory
    decoder_factory(k, symbol_size);
tic(); // Start timing measurements
auto encoder = encoder_factory.build();
auto decoder = decoder_factory.build();
auto recoder = decoder_factory.build(); // Recoding profile only
kodo::set_systematic_off(encoder);
std::vector<uint8_t> data_in(k * symbol_size);
std::vector<uint8_t> payload(encoder->payload_size());
auto data = sak::storage(data_in);
encoder->set_symbols(data);
while (not decoder->is_complete())
{
    encoder->encode(payload.data());
    recoder->decode(payload.data()); // Recoding profile only
    recoder->recode(payload.data()); // Recoding profile only
    decoder->decode(payload.data());
}
toc(); // Stop timing measurements
```

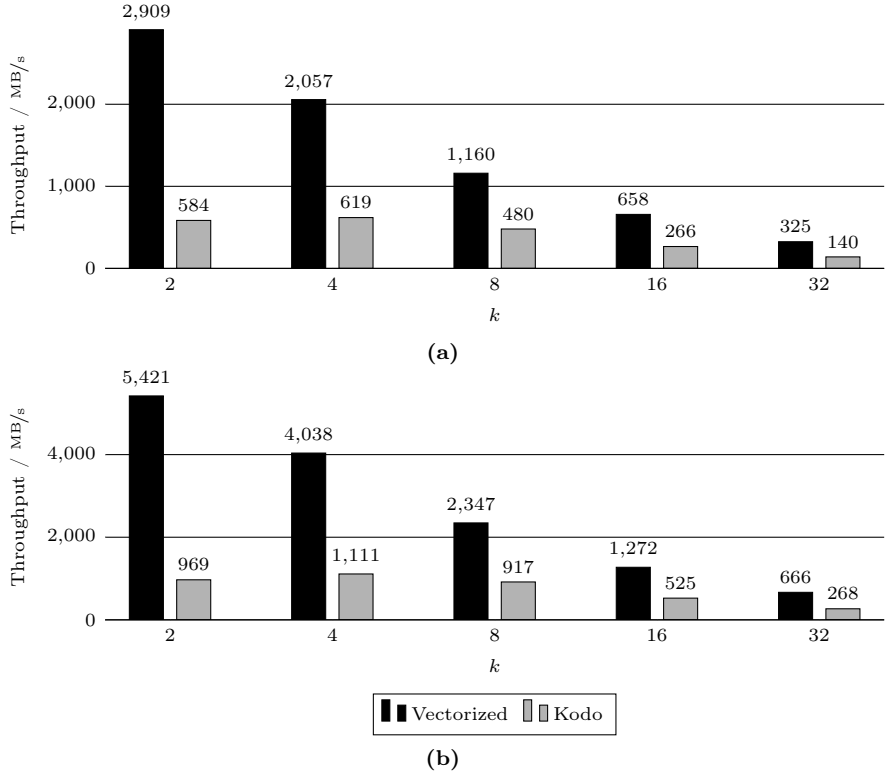



**Fig. A.2:** Coding throughput, on x86 with deep memory, when performing encoding, (for (a) also recoding), and decoding.

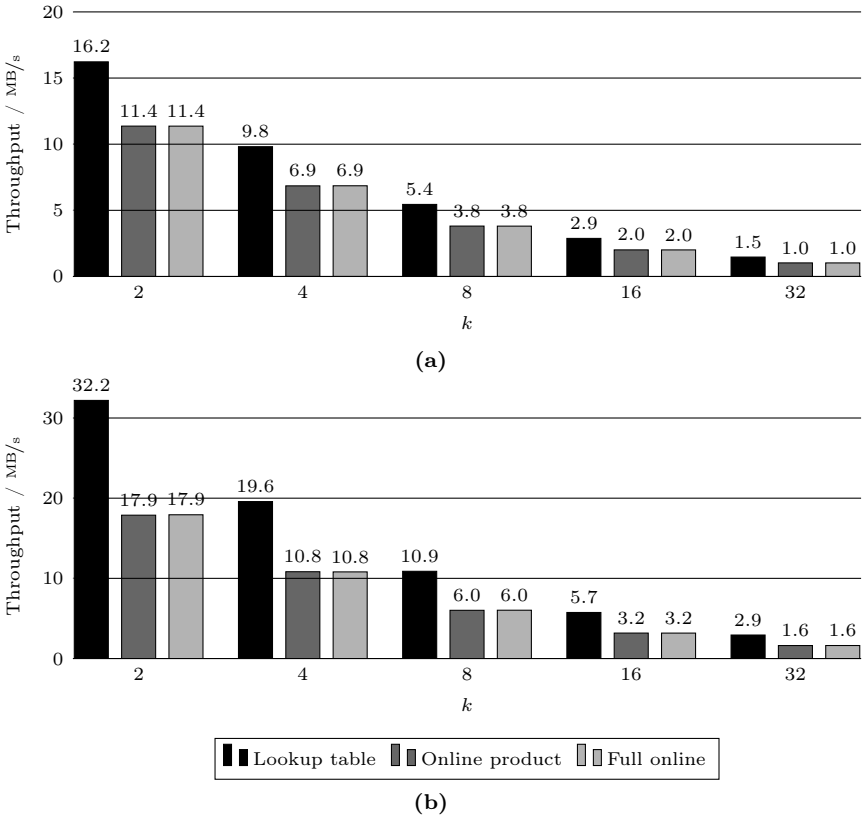



**Fig. A.3:** Coding throughput, on ARM, when performing encoding, (for (a) also recoding), and decoding.

# 4 Coding Throughput of Block Codes on ARM

The ARM processors have become synonymous with modern embedded platforms, and the ARM processors are frequently used in cellular phones, tablets, and set-top-boxes. For that reason, we have designed our software library to utilize the SIMD instructions that ARM processors support. For our experiments we used a Raspberry Pi 2, which is equipped with an quad-core ARMv7 processor running at 900 MHz. The encoding is triangular as shown in the previous section, in Listings A.3 and A.4.

First we consider the throughput performance when SIMD instructions are not enabled. That is, the *Vectorized* implementation is not available and addition operations over the finite field are not vectorized. Fig. A.3a and A.3b show the throughput performance. For the case without recoding it is clear

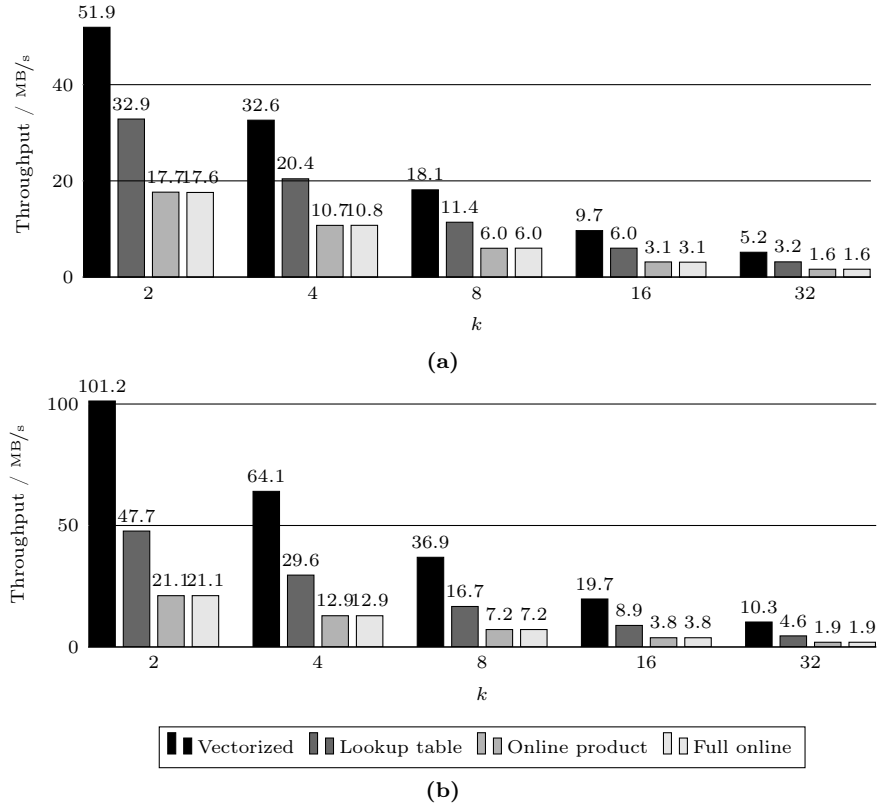



**Fig. A.4:** Coding throughput, on ARM using NEON, when performing encoding, (for (a) also recoding), and decoding.

that the use of a lookup table based implementation of multiplication is the key enabler when maximizing the performance. For the case with recoding, the trend is somewhat similar, however, to a minor extend. With recoding our *Lookup table* implementation is capable of coding in real-time, when $k \leq 8$. Without recoding the same implementation can handle real-time encoding and decoding when $k \leq 16$.

## 4.1 Using NEON

On the ARM platform the SIMD instruction set is denoted NEON. NEON instructions was introduced to the ARMv7, thus they are available in our Raspberry Pi 2. The NEON instructions are capable of operating on at most 128 bytes simultaneously. Fig. A.4a and A.4b show the throughput performance, when NEON instructions are enabled. From the figures it is clear that the primary element to ensure high performance is vectorization of finite field mul-

tiplication, just like on the x86 platform. It is, however, also clear that vectorization of addition operations provide a significant gain, by comparing the performance to when NEON is not enabled. It is clear that the gains of using a lookup table for inversion does not provide any measurable gain. When using the vectorized implementation our software library is capable of maintaining a throughput high enough to perform encoding, recoding, and decoding of high resolution audio in real-time when $k \leq 32$.

# 5 Coding Throughput of Convolutional Codes on x86

In this section we discuss three different methods for storing the coding coefficients for convolutional codes. Software implementation of coders using block codes may allocate (or reclaim from a memory manager) all the needed memory at instantiation. This is due to the fact that the number of coefficients is known exactly at instantiation. This is not the case for convolutional recoders and decoder, and thereby also encoders. In Papers D and E the maximum number of non-zero coefficients from a convolutional encoder and recoder is derived. However, the maximum number of non-zero coefficients in a coding vector during decoding cannot be exactly known at instantiation. There are essentially two ways to overcome this issue. First, use a dynamic array to store the coefficients, and then resize (allocating a new larger segment and copying all the coefficients) when the capacity is depleted. Second, use a fixed size array with a size large enough to hold the largest expected number of non-zero coefficients. The former may suffer from too many reallocations, whereas the latter may throw an exception if the number of non-zero coefficients is too large.

## 5.1 Dynamic Array of Coefficients

For block codes, which has a fixed number of coefficients, a single continues array of coefficients may be used. That is, for block codes a simple array may be used to store the coefficients and the columns they refer to are then the indexes of the array. Since the termination point of a convolutional code is not generally known, it is not possible to use a single array for the coefficients. Thus two arrays are used, one for the non-zero coefficients and one for the column indexes the coefficients refer to. The implementation using dynamic arrays use two instantiations of the template class `std::vector` (part of the C++ standard library) to store the coefficients and columns. This class provide the needed resizing methods. Specifically, the columns use `std::vector<std::uint32_t>` and the coefficients use `std::vector<std::uint8_t>` when the field is GF($2^8$).

When performing vector addition, e.g., $v_1+v_2 = v_3$, then for each of the non-zero coefficients in $v_1$ and $v_2$ the corresponding column index is searched for in

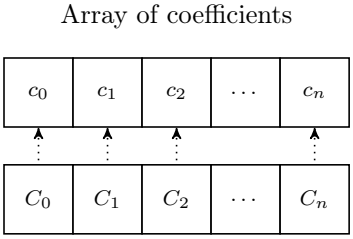



**Fig. A.5:** Mapping between columns and coefficients, when using a fixed size array of coefficients.

the other vector. If the column is found then the addition of the two coefficients is executed, and the result is stored in $v_3$. If the column is not found, then the coefficient is directly stored in $v_3$. Columns that are represented in both $v_1$ and $v_2$ are only handled once. This may be a time consuming search, even if the columns are sorted. On the other hand, when performing multiplication of a vector with a scalar then no search is needed. In that case the multiplication is sequentially performed for each non-zero coefficient.

## 5.2 Fixed Size Array of Coefficients

To avoid the overhead of resizing and moving the coefficients when using a dynamic array, a fixed size array can be used. Fig. A.5 shows the mapping from a column to a coefficient. This method allows for at most $n + 1$ non-zero coefficients, where $n$ is a design parameter. This method follow the same pattern for vector addition and vector/scalar multiplication arithmetic as the method using a dynamic array.

## 5.3 Segmented Array of Coefficients

As mentioned earlier, modern CPUs support vectorized arithmetic through SIMD instructions. In order to utilize this ability a segmented array of coefficients has been implemented. It is segmented in the sense that the array of columns only store the column index of the first coefficient in a block of $N$, where $N$ is the capacity of the SIMD instructions. Fig. A.6 shows the mapping from a column to a coefficient. To simplify the required bookkeeping all the column indexes are a on the form $C_j = iN$, where $j \in \{0, 1, \ldots, n\}$ and $i \in \mathbb{N}_0$. When the column indexes are on this form, they are always "aligned" with the column indexes of another other vector. With respect to vector addition this method can reduce the amount of searches significantly. This is because

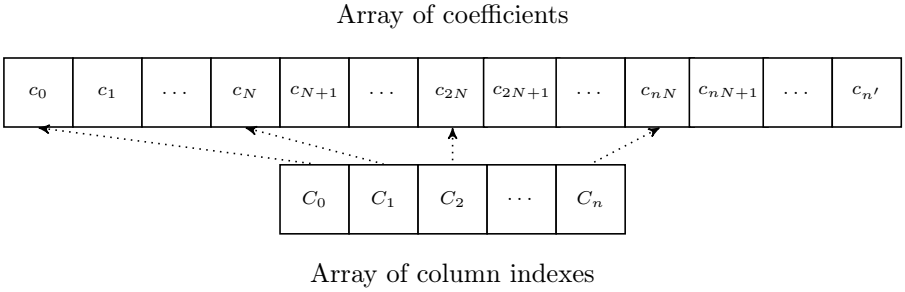



**Fig. A.6:** Mapping between columns and coefficients, when using a segmented array of coefficients.

the columns are grouped together in the segments, and thus fewer columns are searched for. For each segment SIMD operations are used to decrease the computation time.

## 5.4 Performance Evaluation

We now evaluate the performance of the three types of storing the coding coefficients. The evaluation is carried out on an Intel x86 platform, since we are evaluating the difference between the implementations we are only interested in the relative throughput differences. We use two forms of evaluation, namely coding throughput and coefficient throughput. The two cases are evaluated for $k \in \{2, 4, 8, 16, 32\}$ by encoding, recoding, and decoding, of symbols consisting of 1600 and 32 bytes, respectively.

The coding throughput show case the throughput the software implementation is capable of when used for coding a realistic symbol size for network applications. Figures A.7a and A.7b show the coding throughput with and without recoding, respectively. It is interesting to see that the type of coefficient storage that performs best depends on $k$. Furthermore, at no time is the use of a dynamic array the best performing solution.

Due to the increased bookkeeping the throughput of the convolutional codes do not reach the same level as the throughput of our implementation of block codes. However, for all tested values of $k$ the throughput of all four implementations is well above the required 4.608 $^{\text{MB}}/_{\text{s}}$, as expected.

The coefficient throughput is of interest in order to remove the dependency on coding of the data. Figures A.8a and A.8b show the coefficient throughput with and without recoding, respectively. The performance results of the coefficient throughput show similar trends as the coding throughput, but the differences have been magnified, due to the minimalistic symbol size. The biggest difference between coding and coefficient throughput is for $k = 2$, where there

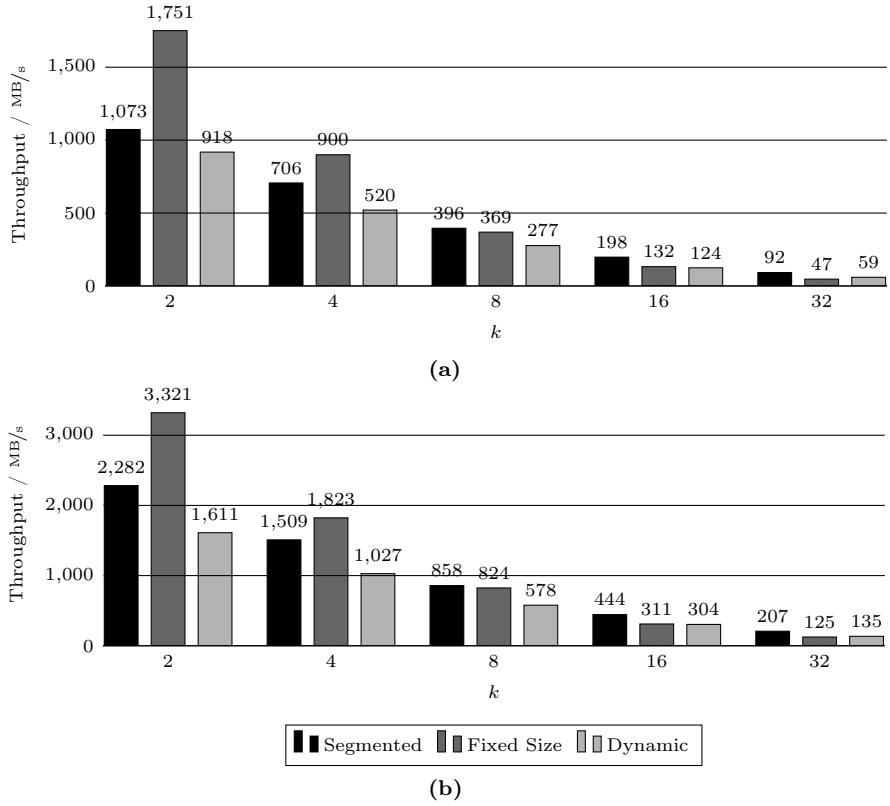



**Fig. A.7:** Coding throughput, on x86, when performing encoding, (for (a) also recoding), and decoding.

is a slight change in ranking when recoding is included.

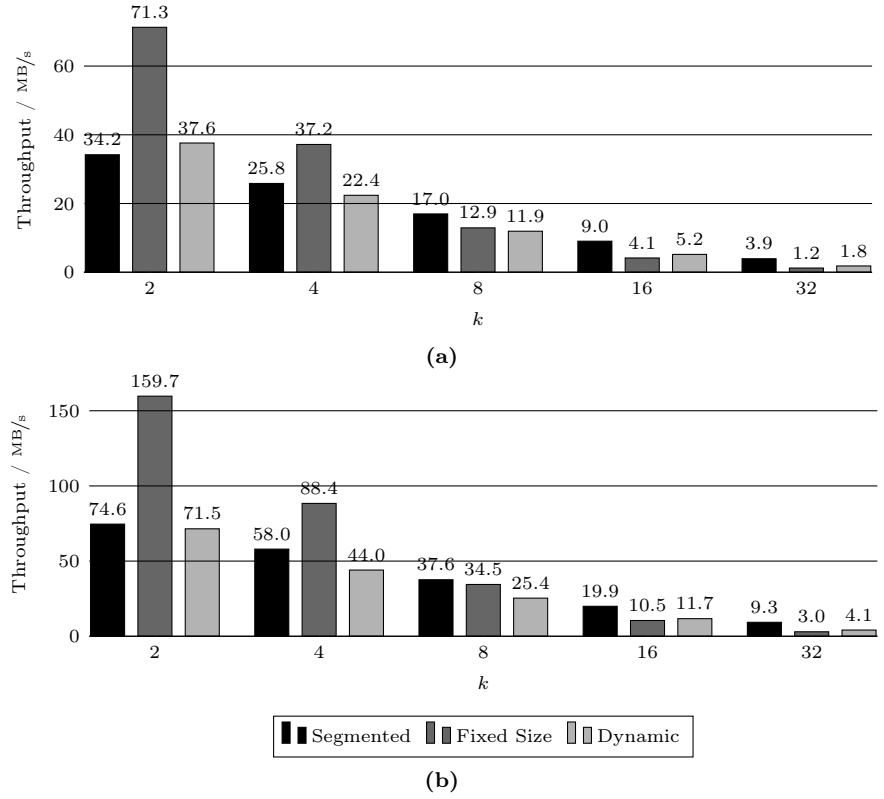



**Fig. A.8:** Coefficient throughput, on x86, when performing encoding, (for (a) also recoding), and decoding.

# 6 Using a Wi-Fi Network for Low Delay Streaming

There are several possible issues when distributing data on an Wi-Fi network. Some of these potential issues will be covered throughout this section. The Figures A.9, A.10, and A.11 show three different casting methods on the same network. On the figures the dotted links are unreliable. That is, there will not be sent any acknowledgements for the packets transmitted on those links. On the other hand, the solid links use both acknowledgements and a number of retransmissions, if required. Furthermore, those links also employ some form of rate adaptation, to increase the probability of successful reception. Both the number of retransmissions and rate adaptation algorithms are implementation specific, and will not be covered further. Finally, the three figures include three types of nodes: source, AP, and sink. The AP is the access point that provides

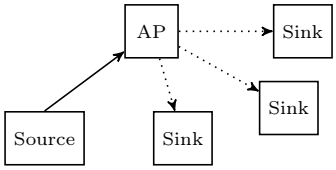



**Fig. A.9:** Unicast from source to AP and broadcast from AP to the three sinks. Dotted links are unreliable.

the wireless network.

## 6.1 Broadcast

Conceptually, using broadcast is compelling, since it allows the source to transmit to several sinks simultaneously. Figure A.9 illustrates how packets are broadcasted on Wi-Fi networks operating in infrastructure mode. However, there are multiple issues associated with broadcasting.

- Broadcast and multicast packets cannot be acknowledge on the physical layer by any of the potential receivers [10]. That is, using broadcast or multicast is unreliable in the sense that the source is unaware of any erasures. Furthermore, without the acknowledgement mechanism these packets are also not retransmitted in case of erasures.

- As it is illustrated in Fig. A.9, it is the AP that broadcasts packets, whereas the source uses unicast to the AP, i.e., the packets to be broadcasted are transmitted at least twice on the network.

- In [11, Section 10.2.1.1] the following is stated:

  > "*If any STA in its BSS is in PS mode, the AP shall buffer all group addressed BUs and deliver them to all STAs immediately following the next Beacon frame containing a DTIM transmission.*"

  That is, if any node (STA) in the network (BSS) is in power save (PS) mode, then the access point (AP) shall transmit all broadcast and multicast packets (BUs) after a special broadcast beacon frame. For the setup described in Section 7, the beacon interval is ∼100 ms, and as a consequence, low-latency streaming is impossible via broadcast or multicast if one of the nodes in the network is in power save mode.

- The data rate used when transmitting broadcast or multicast packets is implementation defined [12]. That is, the manufacturer of a wireless network interface cards (WNIC) is allowed to pick any of the valid bit rates, to be used for broadcast or multicast transmissions. Another effect

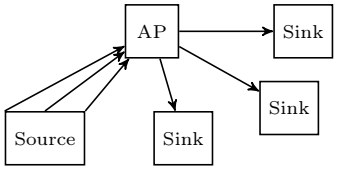



**Fig. A.10:** Unicast from source to AP and from AP to the three sinks.

of this is that the broadcast and multicast packets does not get any of the potential benefits from rate adaptation.

## 6.2 Unicast

Given that broadcast is not a valid option for low delay streaming, as elucidated in the previous section, the alternative could be unicast. While unicast does use an acknowledgement mechanism to retransmit erased packets and adapt the bit rate, it does not scale well for large networks. The problem with scaling is that source has to transmit the entire stream to each sink. The issue of bandwidth consumption is illustrated in Fig. A.10, where both the source and the AP transmit the stream three times, each.

## 6.3 Pseudo Broadcast

The optimal casting-strategy should only transmit the stream once, however, for Wi-Fi networks operating in infrastructure mode all nodes have to communicate through the AP. Thus, it is acceptable to have the source node transmit to the AP and then have the AP distribute reliably it to all the sinks. One way of achieving this is by using pseudo broadcast. As shown in Fig. A.11, with pseudo broadcast the source unicast the stream to the AP and that the AP then unicast to only one of the sinks. The remaining sinks then receive the stream by overhearing the unicast stream, even though it is not addressed to them. Any missing packets may be retransmitted, when one or more sinks signals a negative acknowledgement, on the application layer.

There are both benefits and drawbacks of pseudo broadcast. The primary benefit is that it may significantly reduce the bandwidth requirements, compared to pure unicast. This also enables pseudo broadcast to be highly scalable in number of receivers. Furthermore, by utilizing a forward erasure correcting code retransmissions may be replaced by coded packets which can then be of benefit for more than one sink. Finally, since the stream is transmitted using unicast any retransmission, either between the source and the AP or between the AP and the sink, may also be overheard by the sinks.

The drawbacks are mainly focused around support. That is, for pseudo broadcast to work, overhearing must be enabled and supported by the hard-

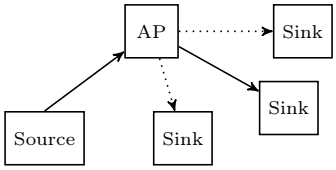



**Fig. A.11:** Unicast from source to AP and unicast from AP to one of the sinks. Dotted links are unreliable.

ware, the firmware, and the software. To the best of the authors knowledge there are no manufacturers of WNICs that officially support overhearing, and thereby also pseudo broadcast. There are though, WNICs that do support it unofficially. Furthermore, Wi-Fi encryption may be troublesome, this issue may be mitigated by using encryption at a higher layer.

## 6.4 Packet Aggregation

In 802.11n and 802.11ac packet aggregation was introduced and extended, respectively. Packet aggregation is somewhat troublesome for erasure correcting codes, since the systematic and coded packets may get aggregated together. This issue is two-fold. First, if a Wi-Fi block is discarded then both the systematic and coded packets are discarded together. Thus, the coded packets serve no purpose, since the systematic and coded packets are collectively either received or discarded. Second, the low delay property of codes with a triangular or convolutional structure is not utilizes since the packets are not sent and received continuously, but in Wi-Fi blocks.

On a more practical level packet aggregation makes it impossible to receive packets using the de facto packet capture library *libpcap* and a transport protocol filter. That is, if the filter is set to e.g., `udp`, then aggregated packets are not delivered by libpcap even though their transport protocol match the filter. If the filter is left empty, then libpcap delivers the aggregated packets. However, they are then deliver in aggregated form. Thus, some mechanism to extract the individual packets from the aggregated packets must be used.

## 6.5 Retransmissions

For unicast transmissions, if a packet is not acknowledge by the receiver the transmitter will retransmit the packet after some delay. The size of the delay depends on the success rate of previous transmissions. For a single packet the maximum number of retransmissions is 7, using the nodes described in Table A.3. This may be an too excessive amount of retransmissions, where in some cases it would be more advantageous to not retransmit the exact same packet, but instead transmit a new coded packet.

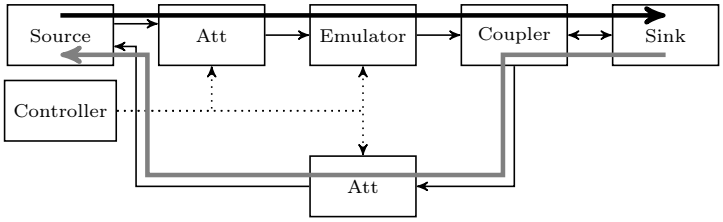



**Fig. A.12:** Experimental setup.

# 7 Experimental Wi-Fi Setup

In this section we present an experimental Wi-Fi testbed with two nodes. To ensure that the experiments are repeatable and consistent, and to eliminate external noise sources, the nodes are connected with wires as opposed to using antennas. The two nodes are described in Table A.3. The nodes are connected using a IEEE 802.11g network in IBSS mode, via the wired connections. The setup is shown in Fig. A.12, where it can be seen that the two nodes (source and sink) communicate directly. The source can transmit high-speed data to the sink (black flow), however, the link from the sink to the source (grey flow) is only able to carry beacons and other management frames. That is, the source broadcasts packets using the highest data rate (54 $^{\text{Mb}}/_{\text{s}}$) and the sink uses the lowest data rate (6 $^{\text{Mb}}/_{\text{s}}$). The use of broadcasting ensures maximum capacity and disables any retransmissions, since broadcast packets are never retransmitted. The *Att*s are attenuators and the *Coupler* is a clockwise directional coupler. Finally, the channel emulator is a EB Propsim C2.

## 7.1 Experimental Procedure

The source is setup to transmit a numbered packet of 1485 bytes to the sink once every millisecond. This procedure is run for 3 hours, and then after $10.8 \times 10^6$ packets the sink reports which packets were received. This packet size was chosen since for a realistic application it would minimizes the protocol over-

**Table A.3:** Description of the nodes used in the experimental setup.

| **Type** | **Name** |
|---|---|
| CPU | Intel Core i3-4170 |
| OS | Ubuntu Linux 15.10 |
| Kernel | Linux 4.4.11 |
| WNIC | Compex WLE600VX |
| WNIC driver | ath10k |
| WNIC firmware | 10.1.467 api 2 |

**Table A.4:** Description of the nodes used in the experimental setup.

| **Name** | **Time** [$\mu$s] |
|---|---|
| DIFS | 28 |
| OFDM Header | 20 |
| Packet header | 20 |
| Data | 240 |
| SIFS | 10 |
| ACK | 34 |
| Backoff | 70 |

head. The packet transmission frequency was chosen in order to utilise about half of the available application bandwidth, in order to avoid congestion. The available application bandwidth is calculated as the amount of data, in bits, divided by total time between two data transmissions. Table A.4 lists the length of each period which a packet transmission consists of [10–13], and the time between them. The random backoff time is calculated as the mean of the minimum random backoff period distribution. In case of erasures or other kinds of disturbance, the backoff period distribution is changed to allow for a larger random backoff, this effect is not included. The sum of the timings in Table A.4 is 422 $\mu$s, thus the maximum available application bandwidth is $\frac{1485 \cdot 8\text{b}}{422\ \mu\text{s}} = 28.2\ {}^{\text{Mb}}/_{\text{s}}$, when transmitting the data with 54 ${}^{\text{Mb}}/_{\text{s}}$.

The channel emulator is used attenuate the signal, and thereby change the signal–to–noise ratio (SNR). The above described test was run for several different levels of SNR. Due to the non-linearity of packet loss as a function of SNR the average packet loss in the experiments are mainly clustered around the extremums, this effect will be elucidated in the following section.

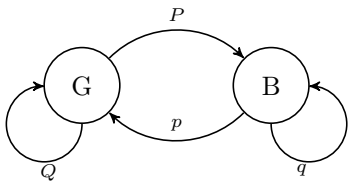



**Fig. A.13:** The Gilbert-Elliott model, an two state Markov model.

# 8 Modelling Burst Losses

When the channel is memoryless, i.i.d. erasures can be modelled using a Bernoulli distribution. When the channel does have memory, it enables a plethora of different channel models. In this section we first cover two widely used channel models, namely the Gilbert-Elliott model [1, 2] and the extended Gilbert model [3]. Finally, we present a new channel model, which is based on the extended Gilbert model.

When $l, l \in \mathbb{N}$ consecutive packets are lost it is termed a burst loss, of length $l$. The probability of the channel experiencing a burst loss of $l$ packets is $f_L(l; P)$. Where $P$ is the state transition matrix of the channel model. That is, $f_L(l; P)$ is a probability mass function termed the burst length distribution.

## 8.1 The Gilbert-Elliott Model

The Gilbert-Elliott model [1, 2] is a two state Markov model, with a state diagram shown in Fig A.13. The two states are denoted the good state (G) and bad state (B), respectively. The probability of successful transmission is in the good state $k$ and $h$ in the bad state. One of the benefits of this simple model is that it can simulate burst losses of any length. However, there is little control to ensure a specific burst length distribution.

## 8.2 The Extended Gilbert Model

The model proposed in [3] is based on an N-state Markov model, and we refer to this model as the the extended Gilbert model. The state diagram for the extended Gilbert model is shown in Fig. A.14 and the state transition matrix is shown in (A.1). The extended Gilbert model provide a high level of control over the burst length distribution. That is, the burst length distribution can be designed directly through $P$. One drawback of this model is that it cannot simulate burst length above $N-1$, i.e., $f_L(l'; P) = 0$ where $l' \in \{N, N+1, \dots\}$.

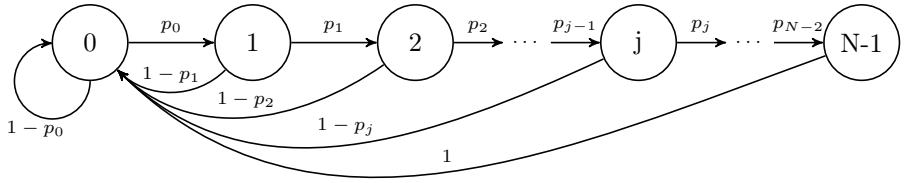



**Fig. A.14:** The extended Gilbert model, an N-state Markov model.

$$P = \begin{bmatrix} 1-p_0 & p_0 & 0 & \cdots & 0 \\ 1-p_1 & 0 & p_1 & \cdots & 0 \\ \vdots & & & \ddots & \\ 1-p_{N-2} & 0 & 0 & \cdots & p_{N-2} \\ 1 & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{A.1}$$

Let $\boldsymbol{\pi}$ be a row vector that is defined by $\boldsymbol{\pi} = \boldsymbol{\pi}P$, that is $\boldsymbol{\pi}$ is the stationary distribution of $P$, where $P$ is given in (A.1). The authors of [3] derive a set of $N$ equations which may be used to determine $\boldsymbol{\pi}$. Moreover, they show that the average erasure probability is $1-\boldsymbol{\pi}_0$, given $P$. The authors did not present any result on the average burst length, to this end we derive this in the following two lemmas.

**Lemma A.1.** *The burst length distribution is a probability mass function, given by*

$$f_L(l;P) = \Pr(L=l;P) = \frac{\boldsymbol{\pi}_l(1-p_l)}{\sum_{j=1}^{N-1} \boldsymbol{\pi}_j(1-p_j)}, \tag{A.2}$$

*where* $L \in \{1,\ldots,N-1\}$.

*Proof.* $\boldsymbol{\pi}_l(1-p_l)$ is the probability of a burst of length $l$ occurring [3], including $l=0$. Thus, normalizing for the range $L \in \{1,\ldots,N-1\}$, yields the probability mass function. □

**Lemma A.2.** *Let* $f_L(l;P)$ *be the probability mass function of the burst lengths given in* (A.2)*, then the average burst length* $\mu_l(P)$ *is*

$$\mu_l(P) = \sum_{j=1}^{N-1} f_L(j;P)j.$$

*Proof.* Each value of $j$ is multiplied with $f_L(j;P)$ and $\mu_l(P)$ is the sum of all these products. □

### List of Obtained Models

Table A.6 on page 65 lists $1 - \boldsymbol{\pi}_0$, $\mu_l(P)$, and the parameters $(p_j)$ for a set of extended Gilbert models. These models were obtained using the method described in section 7.1. The models in Table A.6 are those with $1 - \boldsymbol{\pi}_0 \leq 0.5$ used in [14].

## 8.3 Infinite Hyperbolic Extended Gilbert Model

Consider the Markov chain in Fig. A.14 and let $N \to \infty$, this construction removes the strict maximum burst length of $N - 1$ consecutive packets. We then propose to have $P$ defined with an infinite hyperbolic tail. That is, let the transition probabilities be defined by

$$p_j = \frac{x}{(1+j)^y}, \tag{A.3}$$

where $j \in \mathbb{N}_0$, $0 < x \leq 1$, and $0 < y$. The parameters $x$ and $y$ are denoted the offset and slope parameters, respectively. We define $\boldsymbol{\pi}$ as the stationary distribution of $P$.

**Lemma A.3.** *The stationary distribution must satisfy the following*

$$1 = \sum_{j=0}^{\infty} (\boldsymbol{\pi}_j)\,. \tag{A.4}$$

$$\boldsymbol{\pi}_0 = \sum_{j=0}^{\infty} \boldsymbol{\pi}_j\,(1 - p_j)\,, \tag{A.5}$$

$$\boldsymbol{\pi}_{j+1} = p_j \boldsymbol{\pi}_j. \tag{A.6}$$

*Proof.* It is easy to see that (A.4) holds, since $\boldsymbol{\pi}$ is a distribution all its elements must sum to 1. We have that $\boldsymbol{\pi} = \boldsymbol{\pi} P$, where

$$P = \begin{bmatrix} 1-p_0 & p_0 & 0 & 0 & \cdots \\ 1-p_1 & 0 & p_1 & 0 & \cdots \\ 1-p_2 & 0 & 0 & p_2 & \\ \vdots & \vdots & \vdots & & \ddots \end{bmatrix}.$$

That is,

$$\begin{aligned}\begin{bmatrix}\boldsymbol{\pi}_0 & \boldsymbol{\pi}_1 & \boldsymbol{\pi}_2 & \dots\end{bmatrix} &= \begin{bmatrix}\boldsymbol{\pi}_0 & \boldsymbol{\pi}_1 & \boldsymbol{\pi}_2 & \dots\end{bmatrix}\begin{bmatrix}1-p_0 & p_0 & 0 & 0 & \cdots\\ 1-p_1 & 0 & p_1 & 0 & \cdots\\ 1-p_2 & 0 & 0 & p_2 & \\ \vdots & \vdots & \vdots & & \ddots\end{bmatrix}\\ &= \begin{bmatrix}\boldsymbol{\pi}_0(1-p_0)+\boldsymbol{\pi}_1(1-p_1)+\dots & p_0\boldsymbol{\pi}_0 & p_1\boldsymbol{\pi}_1 & \dots\end{bmatrix}\\ &= \begin{bmatrix}\sum_{j=0}^{\infty}\left(\boldsymbol{\pi}_j\left(1-p_j\right)\right) & p_0\boldsymbol{\pi}_0 & p_1\boldsymbol{\pi}_1 & \dots\end{bmatrix}. \qquad \text{(A.7)}\end{aligned}$$

The two remaining requirements (A.5) and (A.6) are obtained from (A.7). □

Using a method similar to Lemma A.2, we may calculate the average burst length. That is, by calculating the mean of the burst length distribution, i.e.,

$$f_L(l;P) = \Pr(L=l;P) = \frac{\boldsymbol{\pi}_l(1-p_l)}{\sum_{j=1}^{\infty}\boldsymbol{\pi}_j(1-p_j)}, \qquad \text{(A.8)}$$

$$\mu_l(P) = \sum_{j=1}^{\infty} f_L(j;P)j. \qquad \text{(A.9)}$$

Given any $x$ and $y$ then average erasure probability, $1-\boldsymbol{\pi}_0$, and average burst length, $\mu_l$, can be calculated. From (A.4) and (A.6) it is easily seen that $\boldsymbol{\pi}_0$ can be calculated as

$$\boldsymbol{\pi}_0 = \left(1+\sum_{j=1}^{\infty}\left(\prod_{i=0}^{j-1}\frac{x}{(1+i)^y}\right)\right)^{-1} = \left(1+\sum_{j=1}^{\infty}\left(x^j(j!)^{-y}\right)\right)^{-1}.$$

The following Lemma A.4 provide $\mu_l$ as a function of $\boldsymbol{\pi}_0$ and $x$.

**Lemma A.4.** *Let $p_j, j \in \mathbb{N}_0$ be defined as in* (A.3) *and let $\boldsymbol{\pi}$ be defined as described in Lemma A.3, then average burst length $\mu_l(P)$ is*

$$\mu_l(P) = \frac{1-\boldsymbol{\pi}_0}{\boldsymbol{\pi}_0 x}.$$

*Proof.* Combining (A.8) and (A.9) we get

$$\mu_l(P) = \frac{\sum_{j=1}^{\infty} j\boldsymbol{\pi}_j - \sum_{j=1}^{\infty} j\boldsymbol{\pi}_j p_j}{\sum_{i=1}^{\infty}\boldsymbol{\pi}_i - \sum_{i=1}^{\infty}\boldsymbol{\pi}_i p_i}$$

It follows from (A.4) that $\sum_{i=1}^{\infty} \boldsymbol{\pi}_i = 1 - \boldsymbol{\pi}_0$, and similarly for $\sum_{i=2}^{\infty} \boldsymbol{\pi}_i$. Then using (A.6), we may further reduce the denominator:

$$\begin{aligned}
\mu_l(P) &= \frac{\sum_{j=1}^{\infty} j\boldsymbol{\pi}_j - \sum_{j=1}^{\infty} j\boldsymbol{\pi}_j p_j}{1 - \boldsymbol{\pi}_0 - \sum_{i=1}^{\infty} \boldsymbol{\pi}_i p_i} = \frac{\sum_{j=1}^{\infty} j\boldsymbol{\pi}_j - \sum_{j=1}^{\infty} j\boldsymbol{\pi}_j p_j}{1 - \boldsymbol{\pi}_0 - \sum_{i=2}^{\infty} \boldsymbol{\pi}_i} \\
&= \frac{\sum_{j=1}^{\infty} j\boldsymbol{\pi}_j - \sum_{j=1}^{\infty} j\boldsymbol{\pi}_j p_j}{1 - \boldsymbol{\pi}_0 - (1 - \boldsymbol{\pi}_0 - \boldsymbol{\pi}_1)} = \frac{\sum_{j=1}^{\infty} j\boldsymbol{\pi}_j - \sum_{j=1}^{\infty} j\boldsymbol{\pi}_j p_j}{\boldsymbol{\pi}_1} \\
&= \frac{\sum_{j=1}^{\infty} j\boldsymbol{\pi}_j - \sum_{j=1}^{\infty} j\boldsymbol{\pi}_j p_j}{\boldsymbol{\pi}_0 p_0} = \frac{\sum_{j=1}^{\infty} j\boldsymbol{\pi}_j - \sum_{j=1}^{\infty} j\boldsymbol{\pi}_j p_j}{\boldsymbol{\pi}_0 x}
\end{aligned}$$

We, again, use (A.6) to simplify the numerator:

$$\mu_l(P) = \frac{\sum_{j=1}^{\infty} j\boldsymbol{\pi}_j - \sum_{j=1}^{\infty} j\boldsymbol{\pi}_{j+1}}{\boldsymbol{\pi}_0 x} = \frac{\sum_{j=1}^{\infty} j\boldsymbol{\pi}_j - \sum_{j=2}^{\infty} (j-1)\boldsymbol{\pi}_j}{\boldsymbol{\pi}_0 x}$$

Notice that because of (A.4) there exist the following relation:

$$\begin{aligned}
\left(\sum_{j=0}^{\infty} j\boldsymbol{\pi}_j\right) - 1 = \sum_{j=0}^{\infty} (j-1)\boldsymbol{\pi}_j = \left(\sum_{j=2}^{\infty} (j-1)\boldsymbol{\pi}_j\right) - \boldsymbol{\pi}_0 \Rightarrow \\
\sum_{j=2}^{\infty} (j-1)\boldsymbol{\pi}_j = \left(\sum_{j=0}^{\infty} j\boldsymbol{\pi}_j\right) - 1 + \boldsymbol{\pi}_0 = \left(\sum_{j=1}^{\infty} j\boldsymbol{\pi}_j\right) - 1 + \boldsymbol{\pi}_0
\end{aligned}$$

We now use this relation to further reduce the numerator.

$$\begin{aligned}
\mu_l(P) &= \frac{\sum_{j=1}^{\infty} j\boldsymbol{\pi}_j - \left(\left(\sum_{j=1}^{\infty} j\boldsymbol{\pi}_j\right) - 1 + \boldsymbol{\pi}_0\right)}{\boldsymbol{\pi}_0 x} \\
&= \frac{1 - \boldsymbol{\pi}_0}{\boldsymbol{\pi}_0 x}
\end{aligned}$$

□

Given some average erasure probability, $1 - \boldsymbol{\pi}_0$, there is a limit to the possible values of $\mu_l(P)$. The upper bound of $\mu_l(P)$ is the maximum of the average burst length $\mu_U(P)$, and this upper bound is derived in the following lemma.

**Lemma A.5.** *Let $p_j, j \in \mathbb{N}_0$ be defined as in* (A.3)*, then the upper bound of the average burst length $\mu_U(P)$ is*

$$\mu_U(P) = \frac{1}{1-x}.$$

*Proof.* The average burst length is at its maximum when the magnitude of the slope of $P$ is minimized. Thus, we first consider the values of $p_j$ when $y$ goes towards zero

$$p_j = \lim_{y \to 0} \frac{x}{(1+j)^y} \to x.$$

Next we derive the stationary distribution. Since we have that $p_j \to x$, we substitute this into (A.5) and get

$$\begin{aligned} \boldsymbol{\pi}_0 &= \sum_{j=0}^{\infty} \left(\boldsymbol{\pi}_j \left(1 - p_j\right)\right) = \sum_{j=0}^{\infty} \left(\boldsymbol{\pi}_j \left(1 - x\right)\right) \\ &= (1-x) \sum_{j=0}^{\infty} \left(\boldsymbol{\pi}_j\right). \end{aligned} \tag{A.10}$$

To obtain $\boldsymbol{\pi}_0$ we then substitute (A.4) into (A.10)

$$\boldsymbol{\pi}_0 = 1 - x. \tag{A.11}$$

Note that the average erasure probability is $1 - \boldsymbol{\pi}_0 = x$. Finally, we insert (A.11) into the result from Lemma A.4 and get

$$\mu_U(P) = \frac{1-(1-x)}{(1-x)x} = \frac{1}{1-x}.$$

□

Following the proof of the upper bound of $\mu_l(P)$ are two lemmas stating the lower bound of $\mu_l(P)$.

**Lemma A.6.** *Let $p_j, j \in \mathbb{N}_0$ be defined as in* (A.3) *and let $1 - \boldsymbol{\pi}_0 \leq 1/2$ be the average erasure probability, then the lower bound of the average burst length $\mu_L(P)$ is*

$$\mu_L(P) = 1.$$

*Proof.* The average burst length is at its minimum when the magnitude of the slope of $P$ is maximized. Thus, we consider the values of $p_j$ when $y$ goes towards infinity

$$p_j = \lim_{y \to \infty} \frac{x}{(1+j)^y} \to \begin{cases} x & j = 0 \\ 0 & j \neq 0 \end{cases}. \tag{A.12}$$

Note that (A.6) and (A.12) yield that $\boldsymbol{\pi}_j = 0$ when $j > 1$. From (A.4), (A.5), and (A.12) we get the following expression for the stationary distribution

$$\begin{aligned} 1 &= \sum_{j=0}^{\infty} (\boldsymbol{\pi}_j) = \sum_{j=0}^{1} (\boldsymbol{\pi}_j) \\ &= \boldsymbol{\pi}_0 + \boldsymbol{\pi}_1, \end{aligned} \tag{A.13}$$

$$\begin{aligned} \boldsymbol{\pi}_0 &= \sum_{j=0}^{\infty} (\boldsymbol{\pi}_j (1 - p_j)) = \sum_{j=0}^{1} (\boldsymbol{\pi}_j (1 - p_j)) \\ &= \boldsymbol{\pi}_0 (1 - p_0) + \boldsymbol{\pi}_1 (1 - p_1) = \boldsymbol{\pi}_0 (1 - x) + \boldsymbol{\pi}_1 (1 - 0) \\ &= \boldsymbol{\pi}_0 (1 - x) + \boldsymbol{\pi}_1. \end{aligned} \tag{A.14}$$

Using (A.13) and (A.14) we get the stationary distribution

$$\boldsymbol{\pi}_j = \begin{cases} \frac{1}{1+x} & j = 0 \\ \frac{x}{1+x} & j = 1 \\ 0 & j > 1 \end{cases}. \tag{A.15}$$

From (A.15) we get the bound on $1 - \boldsymbol{\pi}_0$ for $x = 1$, that is

$$1 - \boldsymbol{\pi}_0 \Big|_{x=1} = 1 - \frac{1}{1+x} = 1 - \frac{1}{1+1} = \frac{1}{2}.$$

Finally, we insert (A.15) into the result from Lemma A.4 and get

$$\mu_L(P) = \frac{1 - \frac{1}{1+x}}{\frac{1}{1+x} x} = 1$$

□

**Lemma A.7.** *Let $p_j, j \in \mathbb{N}_0$ be defined as in* (A.3) *and let $1 - \boldsymbol{\pi}_0 \geq 1/2$ be the average erasure probability, then the lower bound of the average burst length $\mu_L(P)$ is*

$$\mu_L(P) = \frac{1}{\boldsymbol{\pi}_0} - 1.$$

*Proof.* Let $x = 1$, then

$$p_j = \frac{1}{(1+j)^y}.$$

We insert $x = 1$ into the result from Lemma A.4 and get

$$\mu_L(P) = \frac{1 - \boldsymbol{\pi}_0}{\boldsymbol{\pi}_0} = \frac{1}{\boldsymbol{\pi}_0} - 1;$$

□

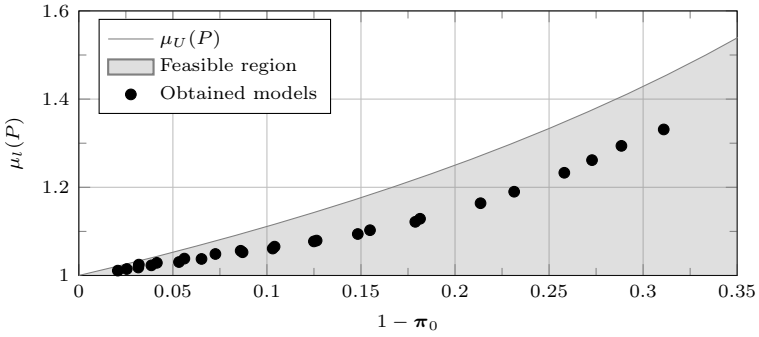



**Fig. A.15:** The average burst length $\mu_l(P)$ as a function of $1 - \boldsymbol{\pi}_0$.

Combining Lemma A.6 and A.7 then gives us

$$\begin{aligned}\mu_L(P) &= \begin{cases} 1 & 1 - \boldsymbol{\pi}_0 \leq 1/2 \\ \frac{1}{\boldsymbol{\pi}_0} - 1 & 1 - \boldsymbol{\pi}_0 \geq 1/2 \end{cases} \\ &= \max\left(1, \frac{1}{\boldsymbol{\pi}_0} - 1\right).\end{aligned}$$

Table A.5 lists the parameters of the proposed model for the observed channel characteristics, similar to those shown in Table A.6. The feasible region for $\mu_l(P)$ as a function of $1 - \boldsymbol{\pi}_0$ is defined as the region where

$$\max\left(1, \frac{1}{\boldsymbol{\pi}_0} - 1\right) = \mu_L(P) < \mu_l(P) < \mu_U(P) = \frac{1}{\boldsymbol{\pi}_0}.$$

Fig. A.15 shows $\mu_U(P)$, the feasible region, along with the models from Table A.5 on the next page (and Table A.6 on page 65). From the figure it is clear that all of the obtained models are within the feasible region of the infinite hyperbolic extended Gilbert model. Finally, Fig. A.16 shows the same as Fig. A.15, but for $0 \leq 1 - \boldsymbol{\pi}_0 \leq 0.85$.

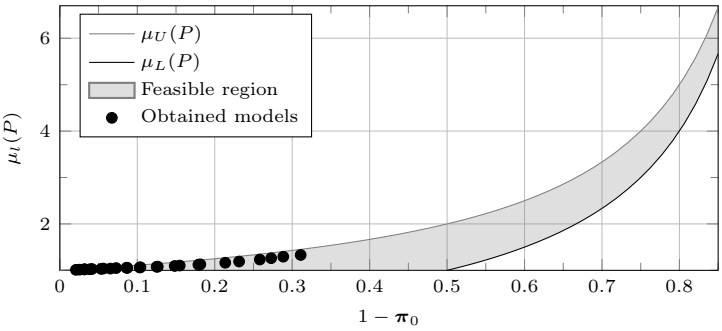



**Fig. A.16:** The average burst length $\mu_l(P)$ as a function of $1-\pi_0$.

**Table A.5:** List of parameters for the proposed model.

| $1-\pi_0$ | $\mu_l(P)$ | $x$ | $y$ |
|---|---|---|---|
| 0.020720 | 1.011019 | 0.020928 | 0.936243 |
| 0.025474 | 1.014671 | 0.025762 | 0.827296 |
| 0.031630 | 1.018042 | 0.032084 | 0.848790 |
| 0.031894 | 1.024648 | 0.032152 | 0.413147 |
| 0.038567 | 1.022981 | 0.039213 | 0.794633 |
| 0.041482 | 1.028690 | 0.042070 | 0.584435 |
| 0.053230 | 1.030480 | 0.054560 | 0.870404 |
| 0.056105 | 1.038282 | 0.057248 | 0.622648 |
| 0.065219 | 1.037489 | 0.067248 | 0.880180 |
| 0.072607 | 1.048741 | 0.074653 | 0.667407 |
| 0.086031 | 1.056043 | 0.089134 | 0.727960 |
| 0.087043 | 1.052686 | 0.090570 | 0.834406 |
| 0.103074 | 1.061083 | 0.108304 | 0.885944 |
| 0.104071 | 1.065411 | 0.109028 | 0.803064 |
| 0.124968 | 1.077267 | 0.132572 | 0.854740 |
| 0.126418 | 1.079169 | 0.134096 | 0.838456 |
| 0.148301 | 1.094038 | 0.159157 | 0.850908 |
| 0.154798 | 1.102705 | 0.166091 | 0.795535 |
| 0.178878 | 1.121662 | 0.194217 | 0.794681 |
| 0.181397 | 1.128626 | 0.196339 | 0.739346 |
| 0.213570 | 1.163882 | 0.233330 | 0.675861 |
| 0.231444 | 1.189946 | 0.253071 | 0.609436 |
| 0.258103 | 1.232916 | 0.282173 | 0.520572 |
| 0.272838 | 1.261601 | 0.297407 | 0.462029 |
| 0.288465 | 1.293943 | 0.313315 | 0.406123 |
| 0.310958 | 1.331163 | 0.339020 | 0.385259 |

# 9 Conclusions

We showed an implementation of a software library for performing network coding. This library is highly flexible and extendable, partly due to the use of the mixin design. For the encoder the flexibility partly lies in the fact that it requires little to no effort to change the behaviour of the coder, e.g., changing from random coefficients to Reed-Solomon codes. For all types of coders it requires little to no effort to change the memory type, e.g., from deep to managed or vice versa.

The software library is capable of encoding and decoding multi channel high resolution audio in real-time. Our x86 implementation is orders of magnitude faster than then the 4.608 $^{\text{MB}}/_{\text{s}}$ requirement, for both block and convolutional codes. We showed three types of implementations of coefficient and column storage for convolutional codes. From our results it is clear that the choice of implementations should depend open the application at hand, e.g., for $k < 8$ the fixed size implementation performed the best and for $k \geq 8$ the segmented implementation performed the best. On an ARM platform the software library is also capable of real-time encoding and decoding of block codes, even without SIMD instructions for $k \leq 16$. We showed that our library is, in some cases, significantly faster than the Kodo version 20.0.0 software library.

We presented a discussion on some of the possible issues of using Wi-Fi, and to some extend how to overcome these issues. We also presented a flexible testbed which has been used to evaluate the performance of erasure correcting codes. The testbed was also used to obtain a set of realistic erasure patterns. We introduced a new channel model, which fits the obtained patterns. These erasure patterns were then modelled using the new channel model and a well known channel model.

**Table A.6:** List of parameters for the extended Gilbert model.

| $1-\boldsymbol{\pi}_0$ | $\mu_l(P)$ | $p_0$ | $p_1$ | ... | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.020720 | 1.011019 | 0.020928 | 0.011019 | | | | | | | | | | |
| 0.025474 | 1.014671 | 0.025762 | 0.014605 | 0.004545 | | | | | | | | | |
| 0.031630 | 1.018042 | 0.032084 | 0.018024 | 0.000992 | | | | | | | | | |
| 0.031894 | 1.024648 | 0.032152 | 0.024568 | 0.003269 | | | | | | | | | |
| 0.038567 | 1.022981 | 0.039213 | 0.022870 | 0.004510 | 0.071429 | | | | | | | | |
| 0.041482 | 1.028690 | 0.042070 | 0.028573 | 0.003857 | 0.062500 | | | | | | | | |
| 0.053230 | 1.030480 | 0.054560 | 0.030324 | 0.004965 | 0.035714 | | | | | | | | |
| 0.056105 | 1.038282 | 0.057249 | 0.038040 | 0.006351 | | | | | | | | | |
| 0.065219 | 1.037489 | 0.067248 | 0.037264 | 0.006048 | | | | | | | | | |
| 0.072607 | 1.048741 | 0.074653 | 0.048223 | 0.009735 | 0.102564 | | | | | | | | |
| 0.086031 | 1.056043 | 0.089134 | 0.055303 | 0.012270 | 0.090452 | | | | | | | | |
| 0.087043 | 1.052686 | 0.090570 | 0.052017 | 0.012335 | 0.041885 | | | | | | | | |
| 0.103074 | 1.061083 | 0.108304 | 0.059982 | 0.017496 | 0.046322 | 0.058824 | | | | | | | |
| 0.104071 | 1.065411 | 0.109028 | 0.063981 | 0.020090 | 0.110619 | 0.020000 | | | | | | | |
| 0.124968 | 1.077267 | 0.132572 | 0.075091 | 0.027233 | 0.063232 | 0.018519 | | | | | | | |
| 0.126418 | 1.079169 | 0.134096 | 0.076295 | 0.033007 | 0.137476 | 0.027397 | | | | | | | |
| 0.148301 | 1.094038 | 0.159157 | 0.089653 | 0.044526 | 0.092916 | 0.060773 | | | | | | | |
| 0.154798 | 1.102705 | 0.166091 | 0.096411 | 0.055456 | 0.168024 | 0.044053 | 0.150000 | 0.666667 | | | | | |
| 0.178878 | 1.121662 | 0.194217 | 0.112391 | 0.072236 | 0.130015 | 0.084158 | 0.098039 | | | | | | |
| 0.181397 | 1.128626 | 0.196339 | 0.116893 | 0.082457 | 0.203514 | 0.060793 | 0.115942 | | | | | | |
| 0.213570 | 1.163882 | 0.233330 | 0.145000 | 0.109379 | 0.169132 | 0.102709 | 0.203297 | 0.135135 | | | | | |
| 0.231444 | 1.189946 | 0.253072 | 0.161707 | 0.134968 | 0.255660 | 0.117481 | 0.217865 | 0.200000 | 0.100000 | 0.500000 | 1.000000 | 1.0 | |
| 0.258103 | 1.232916 | 0.282173 | 0.193509 | 0.158549 | 0.234020 | 0.158751 | 0.285215 | 0.232653 | 0.087719 | | | | |
| 0.272838 | 1.261601 | 0.297408 | 0.208489 | 0.186694 | 0.290457 | 0.188253 | 0.269161 | 0.251121 | 0.214286 | 0.125000 | 0.666667 | | |
| 0.288465 | 1.293943 | 0.313315 | 0.228217 | 0.205663 | 0.314317 | 0.198986 | 0.280136 | 0.289394 | 0.146597 | 0.107143 | 0.333333 | 1.0 | |
| 0.310958 | 1.331163 | 0.339019 | 0.253699 | 0.220507 | 0.291721 | 0.218959 | 0.334443 | 0.302488 | 0.157895 | 0.208333 | 0.200000 | 0.5 | |

# References

[1] E. N. Gilbert, "Capacity of a burst-noise channel," *The Bell System Technical Journal*, vol. 39, no. 5, pp. 1253–1265, September 1960.

[2] E. O. Elliott, "Estimates of error rates for codes on burst-noise channels," *The Bell System Technical Journal*, vol. 42, no. 5, pp. 1977–1997, September 1963.

[3] V. K. Varma, "Testing speech coders for usage in wireless communications systems," *IEEE Workshop on Speech Coding for Telecommunications*, pp. 93–94, 1993.

[4] ISO/IEC 14882:2017, "Programming Language — C++," December 2017.

[5] D. A. Moon, "Object-oriented programming with flavors," *ACM Conference on Object-oriented Programming Systems, Languages and Applications*, pp. 1–8, October 1986.

[6] M. VanHilst and D. Notkin, "Using C++ templates to implement role-based designs," *Object Technologies for Advanced Software: Second JSSST International Symposium*, pp. 22–37, March 1996.

[7] I. S. Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[8] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," *IEEE International Symposium on Information Theory*, June 2003.

[9] M. V. Pedersen, J. Heide, and F. H. P. Fitzek, *Kodo: An Open and Research Oriented Network Coding Library.* Springer Berlin Heidelberg, 2011, pp. 145–152.

[10] E. Perahia and R. Stacey, *Next Generation Wireless LANs: 802.11n and 802.11ac*, 2nd ed. Cambridge University Press, 2013.

[11] ISO/IEC/IEEE 8802-11:2012(E) (Revison of ISO/IEC/IEEE 8802-11-2005 and Amendments), "Information technology — Telecommunications and information exchange between systems Local and metropolitan area networks — Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," pp. 1–2798, November 2012.

[12] ISO/IEC 8802-11:2005/Amd.4:2006(E) IEEE Std 802.11g-2003 (Amendment to IEEE Std 802.11-1999), "Information technology — Local and metropolitan area networks — Part 11: Wireless LAN Medium Access Control (Mac) and Physical Layer (PHY) Specifications-Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band," pp. 1–83, August 2006.

[13] J. Jun, P. Peddabachagari, and M. Sichitiu, "Theoretical maximum throughput of IEEE 802.11 and its applications," *IEEE International Symposium on Network Computing and Applications*, pp. 249–256, April 2003.

[14] J. Hansen, J. Østergaard, J. Kudahl, and J. H. Madsen, "Superregular lower triangular toeplitz matrices for low delay wireless streaming," *IEEE Transactions on Communications*, vol. 65, no. 9, pp. 4027–4038, September 2017.

# Paper B

## Superregular Lower Triangular Toeplitz Matrices for Low Delay Wireless Streaming

Jonas Hansen, Jan Østergaard, Johnny Kudahl, and John H. Madsen

The paper has been published in the
*IEEE Transactions on Communications*, Vol. 65, no. 9, pp. 4027–4038, 2017.

*The layout has been revised.*

# Abstract

*A matrix is termed superregular if all of its possible submatrices are non-singular. Superregular lower triangular Toeplitz matrices are useful for MDS convolutional codes and (sequential) network codes. In this work we present explicit matrix constructions for superregular lower triangular Toeplitz matrices in* $\mathrm{GF}(2^p)^{k\times k}, k \leq 5$. *For* $k > 5$ *we provide a greedy algorithm which (over sufficiently large fields) is guaranteed to find a superregular lower triangular Toeplitz matrix. We introduce (product preserving) joint superregularity, and extend our explicit matrix constructions to these cases. We provide methods for deriving the exact symbol loss probability and delay for any deterministic block code. We derive the exact symbol loss probability and delay for codes using a superregular lower triangular matrix and for codes using two (product preserving) jointly superregular lower triangular matrices. We then compare these results with those obtained from both simulations and our practical implementation, and for each case we also compare with random based codes. Furthermore, our experiments show a gain in coding throughput above 40 % for superregular lower triangular Toeplitz matrices over random matrices.*


# 1 Introduction

The demand for low latency streaming of video and audio is ever increasing [1]. The combination of high resolution video and audio streaming to multiple receivers with low latency is a challenge even for modern wireless networks [2]. To this end, a first step may be to use some form of forward erasure correction (FEC) code. One benefit of a FEC code is that it allows a trade-off between bandwidth, latency and loss probability [3]. By the same token, carefully designing the FEC code together with a transport protocol to fit the application and network in question can significantly improve the user experience [4]. For streaming applications with strict play-out timing constraints such as music streaming, a FEC code can even help in reducing the play-out buffer size at the decoder [5]. However, the joint design of a FEC code and transport protocols is non-trivial. Thus, a lot of research efforts have gone into this area. In [6], the authors propose a combination of Hybrid ARQ and FEC in a transmission protocol based on acknowledgments for reliability. However, the use of acknowledgments in a one-to-many streaming scenario is not desirable. Nevertheless, the method of using both ARQ and FEC still has a lot of potential.

Erasure correcting codes in general can be used in a continuous manner (convolutional codes [7]) or as sequential block codes [8]. For both types decoding may be performed when a packet is received from the network. This may, depending on the code design, help to keep the latency as low as possible.

Encoding is performed at the source and if the code is convolutional or a block code with triangular structure then encoding may also be performed continuously. This is contrary to encoding with a dense block code, where encoding cannot be performed until the entire dataset has been collected.

If additional encoding is performed within the network (and not only at the source) it is usually referred to as network coding [9]. Network coding may increase both reliability by reducing the end-to-end loss probability and the robustness of transport protocols [10, 11]. In addition, network coding can offer increased throughput and has been successfully studied and applied in various communication scenarios. In [12] a middle node combines two or more flows using XOR and broadcasts the result. This method can significantly increase the network efficiency, if the erasure probability does not exceed $\sim$0.1 (with 2 receivers), in which case the algorithm reverts to forwarding. In [13] the authors propose a routing algorithm where neighboring nodes assist (through overhearing and recoding) the transmission between other nodes.

Some codes use random numbers as the coding coefficients. This class of codes is usually termed random linear codes [14]. Random based network codes inherit some simplicity in terms of design with respect to coordination between nodes when used in routing [15]. However, there exist random based network codes where some coordination is required [16]. For random based codes, optimal decoding capabilities can often be proven at least asymptotically, when the field size and code dimension are sufficiently large [17]. For small code dimensions and non-binary fields, random based codes show, on average, near optimal decoding capabilities [18].

With respect to streaming, small coding dimensions are in fact advantageous. There are two main reasons why small coding matrices are interesting from a practical perspective. First, the receiver may use a general decoding algorithm such as Gaussian elimination. Even though this algorithm has cubic complexity, it is possible to use it on embedded platforms when the coding dimensions are small. Second, the small dimension allows for the construction of coding matrices that are guaranteed to be optimal in the non-asymptotic regime [19]. We strongly encourage the use of binary extension fields on embedded platforms, because of the digital architecture and the reduced computational capabilities. For this reason, it is generally unproblematic to implement $\mathrm{GF}(2^p)$ arithmetic on these platforms [20]. The field $\mathrm{GF}(2^8)$ is especially attractive, since each element of the field fits into a single byte.

It was shown in [21] that any dense superregular matrix can be used to construct a systematic maximum distance separable (MDS) block code. A lower triangular matrix is considered superregular, if and only if all of its proper submatrices are non-singular [22]. The authors of [22] showed that an MDS convolutional code can be constructed from superregular lower triangular Toeplitz matrices. It is therefore of great interest to be able to construct superregular lower triangular Toeplitz matrices in small dimensions and with small field

sizes. In [22], the authors presented a few such matrices. However, no insights as to how they were obtained were included. In [23], an explicit construction for superregular (totally positive) matrices was provided for real and complex fields. This construction can easily be extended to very large prime fields, although these are impractical. In [24], a new class of lower block triangular matrices that are superregular over a sufficiently large field was presented.

Concatenating the identity matrix with $m$ matrices results in a generator matrix that produces a block code with rate $1/(m+1)$. Video and audio streaming applications may benefit from codes with a rate lower than $1/2$, especially if the underlying channel suffers from a high erasure rate, or in a one-to-many scenario such as streaming to multiple surround sound speakers. Unfortunately, even if all $m$ matrices are superregular, the resulting code is not guaranteed to have optimal decoding capabilities. To this end, we introduce the notion of jointly superregular matrices. The use of two jointly superregular matrices maximizes the decoding capabilities, see Definition B.2.

In this paper, we provide explicit constructions for all superregular lower triangular Toeplitz matrices over $\mathrm{GF}(2^p)^{k\times k}$ for the case of $k \leq 5$. For general dimensions, we propose a greedy approach to design superregular lower triangular Toeplitz matrices. We also provide explicit constructions for jointly superregular lower triangular Toeplitz matrices for $k = \{2, 3\}$. We then derive the exact symbol loss probability when using (jointly) superregular lower triangular Toeplitz matrices. We show how this symbol loss probability compares to those obtainable in both simulated environments and when using a real Wi-Fi channel. We also compare these symbol loss probabilities with the performance of random linear network coding (RLNC). From this, it is clear that RLNC in several cases achieves, on average, a symbol loss probability that is only slightly larger. Finally, we show that the coding throughput can be significantly increased by using superregular lower triangular Toeplitz matrices compared to RLNC.

Fig. B.1a and B.1c on page 75 show two tall matrices that can be used to construct systematic codes with rate $1/2$ and $1/3$, respectively. On this form it is easy to identify the individual lower triangular matrices which the tall matrices consist of, e.g., $I_4$ and $A'_4$ in Fig. B.1a. Moreover, when used in a real implementation, coding is performed as shown in Fig. B.1b and B.1d. That is, the rows are simply reordered to exploit the low latency property. Let $n, k, l \in \mathbb{N}$ and let $A$ be an $n \times k$ coding matrix, where $A$ can be in any of the forms shown in Fig. B.1. Naturally this yields a code with rate $k/n$. The message to be encoded using $A$ must be stored in $S$, a $k \times l$ matrix, known as the source data matrix. The result of the encoding process is $C$, an $n \times l$ matrix, i.e., the coded data matrix. Therefore $C = AS$, which shows how the $k$ rows (of length $l$) of the source data matrix are encoded into the $n$ rows (of length $l$) of the coded data matrix.

The lower triangular structure is suitable for streaming applications for two

reasons. First, the encoder can transmit the coded packets (a single packet for rate 1/2 codes) immediately after transmitting each systematic packet, as shown in Fig. B.1b and B.1d. That is, the encoding delay is minimized. Second, the decoder can perform real-time decoding of the incoming packets, which then minimizes the decoding delay. Combining these two features with the optimal decoding capabilities results in a family of block codes that are both advantageous and practical for real-time streaming of audio/video content.

Network recoding is drawing attention from several different areas, e.g. ad-hoc and peer-to-peer networks, such as machine-to-machine communication or Internet of Things. The concept of recoding is essentially matrix multiplication. That is, one matrix is used when encoding and another when recoding. In certain cases the resulting code is then their product. We therefore introduce the notion of product preserving jointly superregular matrices. We provide a few explicit constructions for product preserving jointly superregular lower triangular Toeplitz matrices in small dimensions and over any field $\mathrm{GF}(2^p)$. We then derive the exact symbol loss probability for a few selected cases. The exact symbol loss probability is then compared to the achievable symbol loss probability when simulating i.i.d. erasures.

## 2 Superregular Matrices

In [21], the authors defined a dense matrix to be superregular if and only if every square submatrix is non-singular. A lower triangular matrix is defined in [22, Definition 3.3] to be superregular if and only if every proper submatrix is non-singular. The definition of a proper submatrix follows. Let $A$ be a $k \times k$ lower triangular matrix. Let $A' = A^{j_1,\ldots,j_r}_{h_1,\ldots,h_r}$ be an $r \times r$ submatrix of $A$, where $A'$ is constructed using the rows and columns of $A$ with indices $j_1, \ldots, j_r$ and $h_1, \ldots, h_r$, respectively [22, Definition 3.2]. Then, $A'$ is a proper submatrix of $A$ if and only if $1 \leq j_1 < j_2 < \ldots < j_r \leq k$, $1 \leq h_1 < h_2 < \ldots < h_r \leq k$ and $j_t \geq h_t, \forall t$. This notion of superregularity maximizes the decoding capability [25], when a block code with rate 1/2 is generated using a matrix that is constructed as the concatenation of the identity matrix and a superregular lower triangular matrix.[1] A code with a rate higher than 1/2 is obtained by puncturing. We formally define optimal decoding capabilities for block codes in the following definition.

[1] In [26] the authors use a slightly different notion of superregularity.

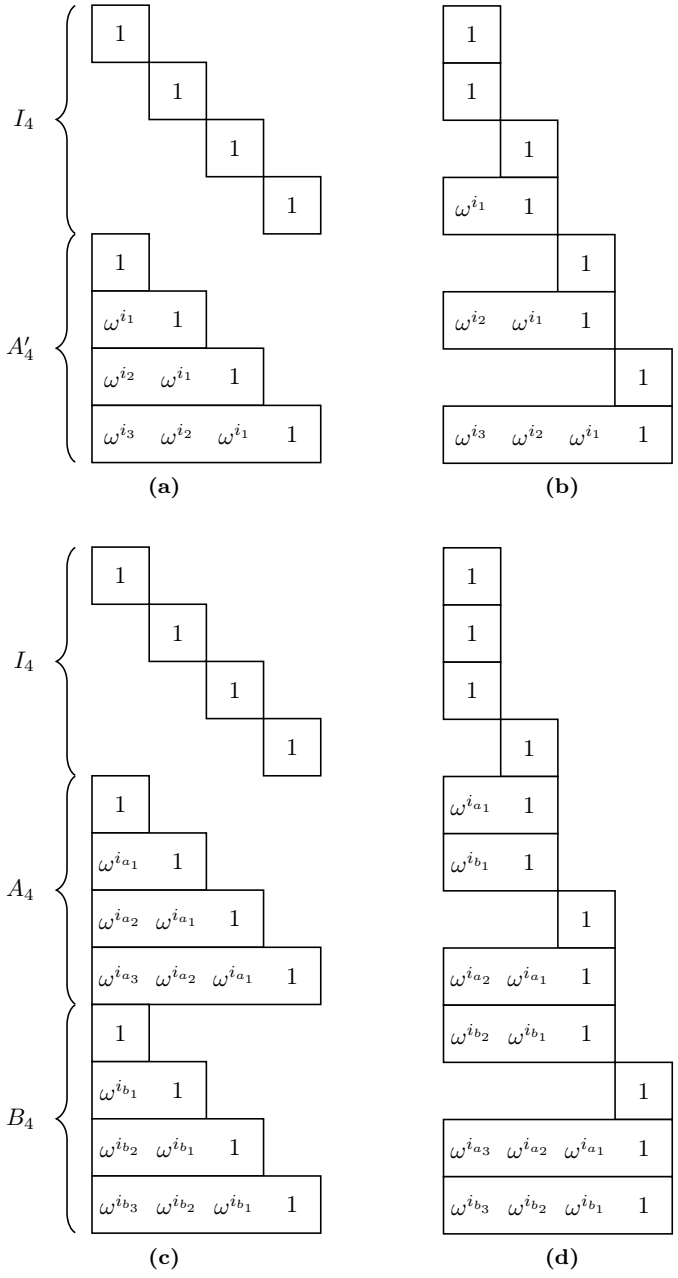



**Fig. B.1:** The matrix structure used in this paper. (a) and (b) are matrices for rate 1/2 codes. (c) and (d) are matrices for rate 1/3 codes. (a) and (c) are used in the lemmas, whereas, (b) and (d) are used on an erasure channel.

**Definition B.1 (Optimal decoding capability).** *A code is said to have optimal decoding capabilities if and only if there does not exist a code (with the same matrix structure) that can recover a higher number of symbols when transmitted over any erasure channel.* △

For optimal decoding capabilities it is a sufficient (but not always necessary) condition that all possible submatrices, the sink can receive, that can be non-singular are non-singular.

**Example B.1.** *Consider the network in Fig. B.2, where the source encodes using the following two matrices:*

$$\begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ b & a & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ h & 1 & 0 \\ i & h & 1 \end{bmatrix},$$

*and the recoder use:*

$$\begin{bmatrix} 1 & 0 & 0 \\ o & 1 & 0 \\ q & o & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ v & 1 & 0 \\ x & v & 1 \end{bmatrix}$$

*for recoding, after partial decoding. The matrices are over* $\mathrm{GF}(2^p)$ *and* $abhioqvx \neq 0$*. If the recoder receives* $[b\ a\ 1]$ *as the first packet it will output two identical packets* $[1\ a/b\ 1/b]$*, this will obviously contain three* $2 \times 2$ *singular submatrices. However, they cannot be made non-singular by changing the code parameters. That is, they are not problematic for achieving optimal decoding capabilities. Let the recoder then receive* $[i\ h\ 1]$ *the output will then be* $\left[o\ 1\ \frac{ao+b+ho+i}{ai+bh}\right]$ *and* $\left[v\ 1\ \frac{av+b+hv+i}{ai+bh}\right]$*. If* $a = \frac{hv+b+i}{v}$*, then* $\frac{av+b+hv+i}{ai+bh} = 0$*, i.e., there is a singular* $1\times1$ *submatrix. If this is the only singular submatrix that could have been non-singular, then the code does indeed recover the maximum possible amount of symbols (no symbol is decodable with this recoder output). That is, the code is optimal, yet not all submatrices that can be non-singular are so.*

A rate 1/3 code can be made from; the identity and two jointly superregular matrices, both individually superregular. The property joint superregularity is described in detail in the coming definition. It fundamentally states that any square submatrix, of all possible square matrices formed by the rows of the two

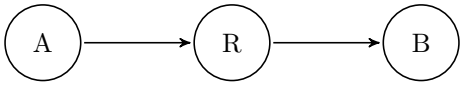



**Fig. B.2:** A multi-hop network with three nodes: a source (A), a relay (R), and a sink (B). R recodes the packets received from A and forwards them to B.

matrices, that can be non-singular must also be so. Following the definition is an example, Ex. 2, which considers two superregular matrices that are not jointly superregular.

**Definition B.2 (Joint superregularity).** *Two superregular $k \times k$ matrices are said to be jointly superregular if and only if all of the proper submatrices of any $k \times k$ matrix, formed by taking $l \in \{1, \ldots, k-1\}$ and $k-l$ rows from the two matrices, respectively, are non-singular. In the context of jointly superregular matrices, a proper submatrix is any square matrix that is not trivially rank deficient. An $m \times m$ matrix, when sorted by increasing row support size*[2]*, is said to be trivially rank deficient if the support of row $i$, $1 \leq i \leq m$, is less than $i$. A proper submatrix need not be triangular.* △

**Example B.2.** *Consider the following two matrices over* $\mathrm{GF}(2^8)$*:*

$$A_3 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 2 & 1 & 1 \end{bmatrix}, B_3 = \begin{bmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 8 & 4 & 1 \end{bmatrix}.$$

*The submatrix constructed from rows* 1 *and* 3 *from* $A_3$ *and row* 3 *from* $B_3$*:*

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 1 \\ 8 & 4 & 1 \end{bmatrix}.$$

*The matrices are not jointly superregular, since the lower-left* $2 \times 2$ *submatrix is singular:*

$$\det\left(\begin{bmatrix} 2 & 1 \\ 8 & 4 \end{bmatrix}\right) = 0.$$

**Definition B.3 (Product preserving joint superregularity).** *Two jointly superregular matrices are product preserving if and only if their product is a superregular matrix.* △

Consider the network presented in Fig. B.2. The matrices $A_k$ and $B_k$ are used for encoding and recoding, respectively. The recoder forwards systematic packets immediately, and recodes every time a coded packet is received. In this setup the product preserving property is a necessary but not sufficient condition for the resulting code to have optimal decoding capabilities. This can easily be realized by considering the case where there are no erasures on the link between A and R. The resulting coding matrix is then $A_k B_k = B_k A_k$, which only has optimal decoding capabilities if it is superregular.

Let $\Omega_{f_p}$ denote the set of roots of a primitive polynomial $f_p$, which generates $\mathrm{GF}(2^p)$. Let $\mathcal{I}_k \triangleq \{i_1, \ldots, i_k : i_j \in \mathrm{GF}(2^p), i_j \neq 2^p - 1, \forall j\}$.

[2]The size of the support of a vector is equal to the number of its non-zero elements.

Let $\psi_\omega(i_1,\ldots,i_{k-1})$ be a $k \times k$ lower triangular Toeplitz matrix with the first column given by $\left[1,\omega^{i_1},\ldots,\omega^{i_{k-1}}\right]^T$. Let $\omega \in \Omega_{f_p}$ and let $\mathcal{A}_k^\omega$ denote the set of all $k \times k$ superregular lower triangular Toeplitz matrices given by $\psi_\omega(i_1,\ldots,i_{k-1})$, where $(i_1,\ldots,i_{k-1}) \in \mathcal{I}_{k-1}$. Let $A_k = \psi_\omega(i_1,\ldots,i_{k-1})$ and let $A_{k+1} = \phi_{\omega,i_k}(A_k) = \psi_\omega(i_1,\ldots,i_k)$ denote the $k+1 \times k+1$ matrix obtained by extending $A_k$.

Let $\mathcal{B}_k^\omega$ be the set of all pairs of jointly superregular lower triangular Toeplitz matrices that satisfy Definition B.2. To increase readability, we add an index to the parameters of the matrices, $(A_k, B_k) \in \mathcal{B}_k^\omega$. That is, $A_k = \psi_\omega\left(i_{a_1},\ldots,i_{a_{k-1}}\right)$ and $B_k = \psi_\omega\left(i_{b_1},\ldots,i_{b_{k-1}}\right)$ are two jointly superregular lower triangular Toeplitz matrices, where $(i_{a_1},\ldots,i_{a_{k-1}},i_{b_1},\ldots,i_{b_{k-1}}) \in \mathcal{I}_{2k-2}$. Let $(A_k,B_k) = \phi_{\omega,i_a,i_b}(A_{k-1},B_{k-1}) = (\phi_{\omega,i_a}(A_{k-1}),\phi_{\omega,i_b}(B_{k-1}))$ be the pair of $k \times k$ matrices obtained by extending $A_{k-1}$ and $B_{k-1}$ using the straightforward generalization of the $\phi$-operator for a single matrix.

The authors of [26] showed a construction of matrices that preserve superregularity after multiplication with block diagonal matrices. Our definition of superregularity does not guarantee that the product of two superregular matrices is superregular. Note that the multiplication (from the right) in [26] is different as the matrices have entries in different fields.

**Lemma B.1.** *Given $A_k \in \mathcal{A}_k^\omega$, then $\exists A'_k \in \mathcal{A}_k^\omega$ such that their product $A_k A'_k \notin \mathcal{A}_k^\omega$.* △

*Proof.* The proof follows easily from [22, Corollary 3.6]. For any $A_k \in \mathcal{A}_k^\omega$ then $A_k^{-1} \in \mathcal{A}_k^\omega$ and it follows that $A_k A_k^{-1} = I_k \notin \mathcal{A}_k^\omega$. □

**Lemma B.2.** *Given $A_k \in \mathcal{A}_k^\omega$, then $\exists B_k, C_k \notin \mathcal{A}_k^\omega$ such that their product $A_k = B_k C_k$, for $k > 1$.* △

*Proof.* Let $A_k = P^{-1}LU \in \mathcal{A}_k^\omega$, where $P^{-1}LU$ is the LU factorization of $A_k$ with partial pivoting and $P$ is a row perturbation matrix. Such a factorization exists for any non-singular matrix, and $P^{-1} \neq I_k$ except for the case $A_2 = \psi_\omega(0)$. Let $B_k = P^{-1}L$ and $C_k = U$, then it follows that neither $B_k$ nor $C_k$ are lower triangular and therefore $B_k, C_k \notin \mathcal{A}_k^\omega$. For the case $A_2 = \psi_\omega(0) \in \mathcal{A}_2^\omega$. Let $B_2 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ and $C_2 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$ then $A_2 = B_2 C_2$ and $B_2, C_2 \notin \mathcal{A}_2^\omega$. □

Let $\mathcal{C}_k^\omega$ denote the set of all pairs of $k \times k$ product preserving jointly superregular lower triangular Toeplitz matrices, according to Definition B.3:

$$\mathcal{C}_k^\omega \triangleq \{(A_k,B_k) \in \mathcal{B}_k^\omega : A_k B_k = B_k A_k \in \mathcal{A}_k^\omega\}, \omega \in \Omega_{f_p}.$$

# 3 Explicit Construction of Superregular and Jointly Superregular Matrices

We begin by showing our explicit construction of superregular lower triangular Toeplitz matrices of size $k \times k$, where $k \leq 5$. All field operations are taken modulo $2^p - 1$, with respect to the elements of $\mathcal{I}_k$. The proof of the following lemma, which states necessary and sufficient conditions for superregularity, is in the Appendix.

**Lemma B.3.** *Let $\omega \in \Omega_{f_p}$ and $A_k = \psi_\omega(i_1, \ldots, i_{k-1})$.*

*i) Then $A_2 \in \mathcal{A}_2^\omega$.*

*ii) Then $A_3 \in \mathcal{A}_3^\omega$ if and only if $(i_1, i_2) \in \mathcal{I}_2$ and $2i_1 \neq i_2$.*

*iii) Let $A_3 \in \mathcal{A}_3^\omega$ and $A_4 = \phi_{\omega, i_3}(A_3)$. Then $A_4 \in \mathcal{A}_4^\omega$ if and only if, $(i_1, \ldots, i_3) \in \mathcal{I}_3$ and satisfy:*

$$3i_1 \neq i_3, \quad i_1 + i_2 \neq i_3, \quad 2i_2 \neq i_1 + i_3. \tag{B.1}$$

*iv) Let $A_4 \in \mathcal{A}_4^\omega$ and $A_5 = \phi_{\omega, i_4}(A_4)$. Then $A_5 \in \mathcal{A}_5^\omega$ if and only if, $(i_1, \ldots, i_4) \in \mathcal{I}_4$ and satisfy:*

$$\begin{aligned} &i_4 \neq 2i_1 + i_2, \quad i_4 \neq i_1 + i_3, \\ &i_4 \neq 2i_2, \quad 2i_3 \neq i_2 + i_4, \quad i_2 + i_3 \neq i_1 + i_4. \end{aligned} \tag{B.2}$$

*and $\omega$ and $(i_1, \ldots, i_4)$ jointly satisfy:*

$$\begin{aligned} 0 \neq &\omega^{2i_2+i_1} \oplus \omega^{i_2+i_3} \oplus \omega^{2i_1+i_3} \oplus \omega^{i_1+i_4}, \\ 0 \neq &\omega^{2i_1+i_4} \oplus \omega^{i_2+i_4} \oplus \omega^{3i_2} \oplus \omega^{2i_3}, \\ 0 \neq &\omega^{2i_1+i_2} \oplus \omega^{i_1+i_3} \oplus \omega^{2i_2} \oplus \omega^{i_4}, \\ 0 \neq &\omega^{2i_1+i_2} \oplus \omega^{4i_1} \oplus \omega^{2i_2} \oplus \omega^{i_4}. \end{aligned} \tag{B.3}$$

△

**Remark B.1.** *Let $\omega \in \Omega_{f_p}$. If $\psi_\omega(i_1, \ldots, i_{k-1}) \in \mathcal{A}_k^\omega$ then $\psi_{\omega'}(i_1, \ldots, i_{k-1}) \in \mathcal{A}_k^\omega, \forall \omega' \in \Omega_{f_p}$.*

Lemma B.4 and B.5 provide necessary and sufficient conditions for constructing jointly superregular lower triangular Toeplitz matrices for $k = 2$ and $k = 3$, respectively. Lemma B.4 also states a necessary condition for constructing jointly superregular lower triangular Toeplitz matrices for any $k > 1$.

**Lemma B.4.** *Let $\omega \in \Omega_{f_p}$. For $k = 2$, $(A_2, B_2) \in \mathcal{B}_2^\omega$ if and only if, $(i_{a_1}, i_{b_1}) \in \mathcal{I}_2$ and $i_{a_1} \neq i_{b_1}$. For any $k > 1$, $(A_k, B_k) \notin \mathcal{B}_k^\omega$, if $\exists j \in \{1, \ldots, k-1\}$ such that $i_{a_j} = i_{b_j}$.* △

*Proof.* Given that $\det\left(\begin{bmatrix} A_{k\,1,j}^{\,j} \\ B_{k\,1,j}^{\,j} \end{bmatrix}\right) = \omega^{i_{a_{j-1}}} \oplus \omega^{i_{b_{j-1}}}$, $\forall 1 < j \leq k$, is only zero when $i_{a_{j-1}} = i_{b_{j-1}}$. □

**Lemma B.5.** *Let* $\omega \in \Omega_{f_p}$, *and let* $(A_2, B_2) \in \mathcal{B}_2^\omega$. *Let* $(A_3, B_3) = \phi_{\omega, i_{a_2}, i_{b_2}}(A_2, B_2)$. *Then* $(A_3, B_3) \in \mathcal{B}_3^\omega$ *if and only if,* $(i_{a_1}, i_{a_2}, i_{b_1}, i_{b_2}) \in \mathcal{I}_4$ *and satisfy:*

$$i_{a_1} + i_{b_1} \neq i_{a_2}, \quad i_{a_1} + i_{b_1} \neq i_{b_2}, \quad i_{a_1} + i_{b_2} \neq i_{a_2} + i_{b_1}$$

*and* $\omega$ *and* $(i_{a_1}, i_{a_2}, i_{b_1}, i_{b_2})$ *jointly satisfy:*

$$\begin{aligned} 0 \neq& \omega^{i_{a_2}} \oplus \omega^{i_{b_2}} \oplus \omega^{i_{a_1}+i_{b_1}} \oplus \omega^{2i_{a_1}}, \\ 0 \neq& \omega^{i_{a_2}} \oplus \omega^{i_{b_2}} \oplus \omega^{i_{a_1}+i_{b_1}} \oplus \omega^{2i_{b_1}}. \end{aligned}$$

△

The proof of Lemma B.5 has been left out given its similarity to the proof of Lemma B.3.

**Lemma B.6.** *Let* $A_k \in \mathcal{A}_k^\omega$, *where* $k > 1$, *then* $(A_k, A_k^{-1}) \notin \mathcal{B}_k^\omega$.

*Proof.* Given that $A_{k1}^2 = \left(A_k^{-1}\right)_1^2 \Rightarrow i_{a_1} = i_{b_1}$ then the requirement of Lemma B.4 is not satisfied. □

For $k = 2$, any pair of jointly superregular lower triangular Toeplitz matrices are also product preserving. Lemma B.7 lists necessary and sufficient conditions for product preserving jointly superregular lower triangular Toeplitz matrices for the case of $k = \{3, 4\}$. The proof of said lemma has been left out because of its similarity to the proof of Lemma B.3.

**Lemma B.7.** *Let* $\omega \in \Omega_{f_p}$.

*i) Let* $(A_3, B_3) \in \mathcal{B}_3^\omega$. *Then* $(A_3, B_3) \in \mathcal{C}_3^\omega$ *if and only if,* $\omega$ *and* $(i_{a_1}, i_{a_2}, i_{b_1}, i_{b_2})$ *jointly satisfy:*

$$0 \neq \omega^{i_{a_2}} \oplus \omega^{i_{b_2}} \oplus \omega^{i_{a_1}+i_{b_1}}, \quad 0 \neq \omega^{i_{a_2}} \oplus \omega^{i_{b_2}} \oplus \omega^{i_{a_1}+i_{b_1}} \oplus \omega^{2i_{a_1}} \oplus \omega^{2i_{b_1}}.$$

*ii) Let* $(A_4, B_4) \in \mathcal{B}_4^\omega$. *Then* $(A_4, B_4) \in \mathcal{C}_4^\omega$ *if and only if,* $\omega$ *and* $(i_{a_1}, \ldots, i_{a_3}, i_{b_1}, \ldots, i_{b_3})$ *jointly satisfy:*

$$\begin{aligned} 0 \neq& \omega^{i_{b_1}+i_{a_3}} \oplus \omega^{i_{b_3}+i_{a_1}} \oplus \omega^{i_{a_1}+i_{a_3}} \oplus \omega^{i_{b_2}+2i_{a_1}} \\ & \oplus \omega^{2i_{b_1}+i_{a_2}} \oplus \omega^{2i_{b_2}} \oplus \omega^{2i_{b_1}+2i_{a_1}} \oplus \omega^{2i_{a_2}} \\ & \oplus \omega^{i_{b_1}+i_{b_3}} \oplus \omega^{i_{b_1}+i_{b_2}+i_{a_1}} \oplus \omega^{i_{b_1}+i_{a_1}+i_{a_2}}, \\ 0 \neq& \omega^{i_{a_3}} \oplus \omega^{i_{b_3}} \oplus \omega^{i_{b_1}+i_{a_2}} \oplus \omega^{i_{b_2}+i_{a_1}} \oplus \omega^{i_{b_1}+2i_{a_1}} \\ & \oplus \omega^{2i_{b_1}+i_{a_1}} \oplus \omega^{3i_{b_1}} \oplus \omega^{3i_{a_1}}, \\ 0 \neq& \omega^{i_{a_3}} \oplus \omega^{i_{b_3}} \oplus \omega^{i_{b_1}+i_{a_2}} \oplus \omega^{i_{b_2}+i_{a_1}}. \end{aligned}$$

△

# 4 Greedy algorithm

In this section we provide a greedy algorithm for constructing an $m \times m$ superregular lower triangular Toeplitz matrix. The algorithm is listed in Algorithm 1 on the following page. The search starts at $k = 1$ with $A_k = 1$ and iterates over all possible values of $i_k$, except the last element. At each step, the matrix $A_k$ is extended using the $\phi$-operator and $i_k$. If the newly extended matrix is found to be superregular, the algorithm increments $k$ and continues to extend the new matrix. The last element, $2^p - 1$, is excluded since $\omega^0 = \omega^{2^p-1}$, where $\omega \in \Omega_{f_p}$. The search continues until the matrix is extended into an $m \times m$ superregular matrix. Backtracking is required if the algorithm reaches $i_k = 2^p - 2, k < m$ and the extended matrix is not superregular. In such a case $k$ is decremented and the search continues with the next $i_k$. No matrix is found if the field size is not sufficiently large. However, if the field size is sufficiently large then the algorithm is guaranteed to find an $m \times m$ superregular lower triangular Toeplitz matrix. Our implementation requires less than 230 ms to find a $9 \times 9$ superregular lower triangular Toeplitz matrix over GF($2^8$), when running on an Intel I5-2415M 2.3 GHz. Our experiments show that backtracking is required to produce a matrix of size $10 \times 10$ over the field GF($2^8$).

$$\mathcal{A}_k(A_k) \triangleq \Big\{ A_k{}^{j_1,\ldots,j_s}_{l_1,\ldots,l_s} \in [\mathrm{GF}(2^p)]^{s\times s} : s = 2,\ldots,k-1, 1 \le j_1 < \ldots < j_s = k, \\ 1 = l_1 < \ldots < l_s \le k, j_t \ge l_t, \forall t \Big\} \tag{B.4}$$

# 5 Theoretical Symbol Loss Probability

Any erasure correcting or network code should be able to recover as many source symbols as possible, i.e., the performance metric is not the rank but the number of decoded source symbols. That is, we consider the probability of losing a source symbol, averaged over all symbols, to be a proper performance metric of a given network code. The reasoning behind this performance metric is that it is only the recovered symbols that are usable, regardless of the rank.

Let $C$ be an $n \times k$ coding matrix over some field GF($q$). The rows of $C$ are the coding vectors for the packets that will be transmitted on the network. A packet transmission may succeed or result in an erasure, and these two outcomes are modeled by 1 and 0, respectively. Let $r$ be the number of links in the logical network. Then the set $V_{n,r} \triangleq \{v_1, \ldots, v_{2^{nr}} \in \{0,1\}^{n\times r}\}$ contains all possible combinations of success and erasure for all independent packet receptions. For example, in a single hop network as shown in Fig. B.3a $r = 1$, and in a multi hop network as shown in Fig. B.3b then $r = 3$. Let $W \in V_{n,r}$ be the indicator matrix which contains a zero at element $(i, j)$ if the $(i, j)$'th packet is erased and a 1 otherwise. We now introduce $T(x, C, W) \in \{0, 1\}$, which denotes the

**Algorithm 1** Greedy search with backtracking for an $m \times m$ superregular lower triangular Toeplitz matrix.

**Input:** $m \geq 2, \omega \in \Omega_{f_p}, A_1 = 1, i_l = 0 \forall l, k = 1$

1: **while** $k < m$ **do**
2:  **while** $i_k < 2^p - 1$ **do**
3:  $A_{k+1} := \phi_{\omega, i_k}(A_k)$
4:  Define $\mathcal{A}_{k+1}$ using (B.4) and $A_{k+1}$
5:  **if** $\nexists A' \in \mathcal{A}_{k+1}$ such that $\det(A') = 0$ **then**
6:  $k := k + 1$
7:  **go to 1**
8:  **end if**
9:  $i_k := i_k + 1$
10:  **end while**
11:  **if** $k = 2$ **then**
12:  **return** *Insufficient field size*
13:  **else**
14:  $i_k := 0$, $i_{k-1} := i_{k-1} + 1$, $k := k - 1$
15:  **go to 2**
16:  **end if**
17: **end while**
18: **return** $A_k$

sink's ability to decode the $x$'th symbol given $C$ and $W$, where $x \in \{1, \ldots, k\}$. That is, $T(x, C, W)$ determines whether or not it is possible for the sink to decode the $x$'th symbol, when receiving the packets from $C$, where successful reception is specified by the elements of $W$.

**Lemma B.8.** *Let $\overline{e} \in [0,1]^r$ be the erasure probability of the $r$ logical links, assuming i.i.d. erasures, and let $W \in V_{n,r}$ be the indicator matrix. Then the probability of $W$ occurring is*

$$P(W|\overline{e}) = \prod_{i=1}^{n} \prod_{j=1}^{r} (1 - e_j) \mathbb{1}[W_{i,j} = 1] + e_j \mathbb{1}[W_{i,j} = 0].$$

△

*Proof.* It follows easily from the i.i.d. assumption that the probability of $W$ occurring is the product of the probabilities of erasure and successful reception on all links. □

**Lemma B.9.** *Let $C \in \mathrm{GF}(q)^{n \times k}$ and $\overline{e} \in [0,1]^r$. Then the average (across all*

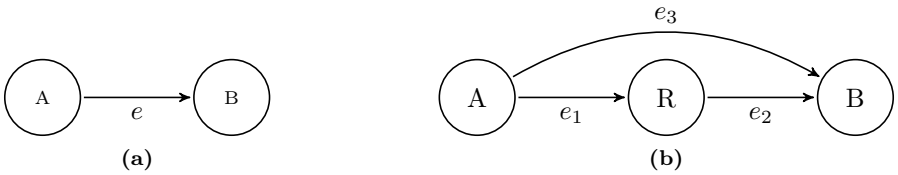



**Fig. B.3:** (a) single hop network with two nodes. (b) recoding network with three nodes, multiple paths and hops. Source (A), Relay (R), and Sink (B).

*symbols) probability of a symbol not being decodable is given by*

$$P_L(C|\overline{e}) = 1 - \underbrace{\frac{1}{k}\sum_{x=1}^{k}\underbrace{\left(\sum_{W\in V_{n,r}} T(x,C,W)P(W|\overline{e})\right)}_{p_1}}_{p_2}. \tag{B.5}$$

△

*Proof.* First $p_1$ sums all the probabilities for combinations of $W$ where the $x$'th symbol is decodable. The result of $p_1$ is the probability that the $x$'th symbol is decodable. Then $p_2$ sums for all symbols and the result is normalized. This gives the average probability that a symbol is decodable. Finally, the probability of decoding a symbol is subtracted from one, to provide the probability that a symbol is not decodable. □

The exact symbol loss probability for any static code, where erasures are assumed to be i.i.d., is then given by $P_L(C|\overline{e})$. Since (B.5) is valid for any static code it is naturally applicable to the three types of codes based on superregular lower triangular Toeplitz matrices. However, it is not applicable to random based codes. $T(x, C, W)$ can be evaluated by trying all combinations, but for a large $nr$ this may become non-trivial.

The following six superregular matrices over GF($2^8$) are used throughout the rest of the paper. When $k$ is smaller than the matrix dimension, e.g., for a superregular $10 \times 10$ matrix such as (B.6) and $k < 10$, then the upper-left $k \times k$ submatrices is used. The matrices (B.8) and (B.9) are jointly superregular,

and (B.10) and (B.11) are also product preserving.

$$A_{10} = \psi_\omega \left(1, 0, 0, 3, 5, 10, 36, 86, 83\right) \tag{B.6}$$
$$A'_{10} = \psi_\omega \left(125, 35, 109, 219, 83, 177, 191, 39, 23\right) \tag{B.7}$$
$$A_7 = \psi_\omega \left(6, 0, 0, 4, 136, 133\right) \tag{B.8}$$
$$A'_7 = \psi_\omega \left(7, 2, 3, 11, 77, 157\right) \tag{B.9}$$
$$A_6 = \psi_\omega \left(0, 2, 5, 0, 15\right) \tag{B.10}$$
$$A'_6 = \psi_\omega \left(1, 0, 4, 9, 30\right) \tag{B.11}$$

## 5.1 Single Hop Network

We now consider a single hop network with two nodes: source A and sink B. The topology is shown in Fig. B.3a. Given that the source transmits data with a rate of $1/2$ and that there is only a single sink, then $r = 1$ and $n = 2k$. Table B.1 lists the polynomial coefficients for the closed-form expressions of (B.5) evaluated for $1 \leq k \leq 10$ using superregular lower triangular Toeplitz matrices, such as (B.6). The polynomials are of degree $n$ and the coefficients for the first and zero orders are equal to zero. The degree is $n$ since the $n$ rows results in $n$ packets transmitted over the link. The theoretical symbol loss probability is shown with solid lines in Fig. B.4a.

For codes with rate $1/3$, then $n = 3k$. Table B.2 lists the polynomial coefficients for the closed-form expressions of (B.5) evaluated for $1 \leq k \leq 7$ using (B.8) and (B.9). The polynomials are of degree $n$ and the coefficients for the second, first and zero orders are zero. The theoretical symbol loss probability is shown with solid lines in Fig. B.4b. Both sets of theoretical symbol loss probabilities are compared to simulations and experimental results in the appropriate sections.

## 5.2 Recoding Network

The second network consists of three nodes, in order to allow for recoding to take place at an intermediate node. The source use (B.10) for encoding and the relay use (B.11) for recoding. Note that our relay only performs recoding if the packet received from the source increases the rank. That is, the relay recodes at most $k$ packets. The topology is shown in Fig. B.3b. Erasures are i.i.d. on the three links. We have considered two sets of erasure probabilities in order to examine the theoretical symbol loss probability of the network. The Sections 5.2 and 5.2, we present the exact symbol loss probability given some $e$, for the case of $k = \{1, 2\}$, in these cases (B.10) and (B.11) have optimal decoding capabilities. For $k \geq 3$ the product preserving property is not sufficient for optimality for the network shown in Fig. B.3b. Thus, the matrices in (B.10) and (B.11) do not achieve optimal decoding capabilities in these cases.

**Table B.1:** Polynomial coefficients for the closed-form expression of (B.5) for $k \leq 10$, $n = 2k$, and $r = 1$ using superregular lower triangular Toeplitz matrices.

| $k$ | $P_L(C\|e)$ coefficients |
|---|---|
| 1 | $1$ |
| 2 | $-1, \frac{3}{2}, \frac{1}{2}$ |
| 3 | $2, -5, \frac{8}{3}, 1, \frac{1}{3}$ |
| 4 | $-5, \frac{35}{2}, -\frac{39}{2}, 5, 2, \frac{3}{4}, \frac{1}{4}$ |
| 5 | $14, -63, 104, -70, \frac{48}{5}, 4, \frac{8}{5}, \frac{3}{5}, \frac{1}{5}$ |
| 6 | $-42, 231, -\frac{1505}{3}, 525, -\frac{730}{3}, \frac{56}{3}, 8, \frac{10}{3}, \frac{4}{3}, \frac{1}{2}, \frac{1}{6}$ |
| 7 | $132, -858, 2304, -3234, 2430, -837, \frac{256}{7}, 16, \frac{48}{7}, \frac{20}{7}, \frac{8}{7}, \frac{3}{7}, \frac{1}{7}$ |
| 8 | $-429, \frac{6435}{2}, -\frac{20559}{2}, 18018, -18459, \frac{42735}{4},$ $-\frac{11515}{4}, 72, 32, 14, 6, \frac{5}{2}, 1, \frac{3}{8}, \frac{1}{8}$ |
| 9 | $1430, -12155, \frac{134992}{3}, -94380, \frac{365288}{3}, -\frac{292292}{3},$ $45472, -\frac{29854}{3}, \frac{1280}{9}, 64, \frac{256}{9}, \frac{112}{9}, \frac{16}{3}, \frac{20}{9}, \frac{8}{9}, \frac{1}{3}, \frac{1}{9}$ |
| 10 | $-4862, 46189, -194337, 474045, -\frac{3677388}{5}, 746460, -\frac{2436588}{5},$ $\frac{947232}{5}, -\frac{173286}{5}, \frac{1408}{5}, 128, \frac{288}{5}, \frac{128}{5}, \frac{56}{5}, \frac{24}{5}, 2, \frac{4}{5}, \frac{3}{10}, \frac{1}{10}$ |

All the polynomials are of degree $5k$, since the $3k$ rows result in $2k$ packets being transmitted from the source to both the relay and the sink. The relay may then also transmit $k$ packets to the sink, and since erasures are i.i.d., the degree is the sum of transmitted packets, where the source's packets are counted twice as they are transmitted on two logical links. The theoretical symbol loss probabilities are compared to that obtained by simulations in Section 7.

**Table B.2:** Closed-form expression of (B.5) for $k \leq 7$, $n = 3k$, and $r = 1$ using two jointly superregular lower triangular Toeplitz matrices.

| $k$ | $P_L(C\|e)$ coefficients |
|---|---|
| 1 | $1$ |
| 2 | $-2, \frac{5}{2}, 0, \frac{1}{2}$ |
| 3 | $7, -16, \frac{28}{3}, -\frac{4}{3}, \frac{5}{3}, 0, \frac{1}{3}$ |
| 4 | $-30, 99, -110, \frac{93}{2}, -12, 7, -1, \frac{5}{4}, 0, \frac{1}{4}$ |
| 5 | $143, -616, 1001, -752, \frac{1397}{5}, -88, \frac{186}{5}, -\frac{48}{5}, \frac{28}{5}, -\frac{4}{5}, 1, 0, \frac{1}{5}$ |
| 6 | $-728, \frac{7735}{2}, -\frac{24752}{3}, \frac{53755}{6}, -\frac{15820}{3}, \frac{3731}{2}, -\frac{1880}{3}, \frac{1397}{6}, -\frac{220}{3}, 31, -8, \frac{14}{3}, -\frac{2}{3}, \frac{5}{6}, 0, \frac{1}{6}$ |
| 7 | $3876, -24480, 64600, -91824, 75990, -38080, \frac{92515}{7}, -4520, 1599, -\frac{3760}{7}, \frac{1397}{7}, -\frac{440}{7}, \frac{186}{7}, -\frac{48}{7}, 4, -\frac{4}{7}, \frac{5}{7}, 0, \frac{1}{7}$ |

### Equal Erasure Probability

In order to investigate the simplest scenario, the three erasure probabilities are equal, i.e., $e_1 = e_2 = e_3 = e$. Let $n = 3k$, $r = 3$ and the encoding and recoding matrices be product preserving jointly superregular lower triangular Toeplitz. Let $k = 1$, then

$$P_L(C|e) = -e^5 + e^4 + e^3.$$

Let $k = 2$, then

$$P_L(C|e) = -4e^{10} + \frac{23}{2}e^9 - 7e^8 - 5e^7 + 3e^6 + \frac{3}{2}e^5 + \frac{1}{2}e^4 + \frac{1}{2}e^3.$$

The theoretical symbol loss probabilities are shown with solid lines in Fig. B.5a.

### Unequal Erasure Probability

In order to investigate a scenario where recoding is favorable, the erasure probabilities of the links to and from R are decreased, that is $2e_1 = 3e_2 = e_3 = e$. Let $n = 3k$, $r = 3$ and the encoding and recoding matrices be product preserv-

ing jointly superregular lower triangular Toeplitz. For $k = 1$, then

$$P_L(C|e) = -\frac{1}{12}e^5 + \frac{1}{4}e^4 + \frac{1}{3}e^3,$$

and for $k = 2$, then

$$P_L(C|e) = -\frac{1}{36}e^{10} + \frac{17}{96}e^9 - \frac{1}{4}e^8 - \frac{65}{288}e^7 + \frac{3}{16}e^6 + \frac{5}{12}e^5 + \frac{1}{8}e^4 + \frac{1}{6}e^3.$$

The theoretical symbol loss probability is shown with solid lines in Fig. B.5b.

# 6 Theoretical Symbol Delay

We introduce $T_{D_B}(x, C, W) \in \{0, \ldots, \frac{n}{k}(k - x + 1)\}$, which denotes the number of packets transmitted from the encoder, and possibly recoders, that are needed before the $x$'th symbol is decoded at the sink given $C$ and $W$, where $x \in \{1, \ldots, k\}$. We further define $T_{D_B}(x, C, W) = 0$ if the $x$'th symbol is not decodable given $C$ and $W$. We use this definition of symbol delay since it makes it clear that the rate of a code will also have an effect on the delay, and the fact that transmission of redundant packets is not free in terms of bandwidth.

**Lemma B.10.** *Let $C \in \mathrm{GF}(q)^{n\times k}$ and $\overline{e} \in [0, 1]^r$. Then the average (across all decodable symbols) symbol delay is then*

$$S_D(C|\overline{e}) = \frac{\sum_{x=1}^{k}\left(\sum_{W\in V_{n,r}} T_{D_B}(x, C, W)P(W|\overline{e})\right)}{\sum_{x=1}^{k}\left(\sum_{W\in V_{n,r}} T_{L_B}(x, C, W)P(W|\overline{e})\right)}. \tag{B.12}$$

*Proof.* The numerator sums the amount of delay all received symbols endure, weighted by the probability of those delays occurring. The denominator sums the number of decodable symbols weighted by the probability of those symbols being decodable. □

Similar to the $P_L(C|\overline{e})$ for the symbol loss probability, $S_D(C|\overline{e})$ yields the exact symbol delay for any static coding matrix, where erasures are assumed to be i.i.d. Thus, it is also applicable to the three types of codes based on superregular lower triangular Toeplitz matrices, but again it is not applicable to random based codes.

Eq. (B.12) can be evaluated in closed-form for the single hop network ($r = 1$). Table B.3 lists the closed-form expressions of (B.12) evaluated for $1 \leq k \leq 4$ using a superregular lower triangular Toeplitz matrix, i.e., $n = 2k$. When using two jointly superregular lower triangular Toeplitz matrices ($n = 3k$) the closed-form expressions of (B.12) for $k = 1$ is

$$S_D(C|e) = \frac{3e^2 + 2e + 1}{e^2 + e + 1},$$

**Table B.3:** Closed-form expression of (B.12) for $k \leq 4$, $n = 2k$, and $r = 1$ using a superregular lower triangular Toeplitz matrix.

| $k$ | $S_D(C\|e)$ |
|---|---|
| 1 | $\frac{2e+1}{e+1}$ |
| 2 | $2\frac{3e^3-2e^2-2e-1}{2e^3-e^2-2e-2}$ |
| 3 | $\frac{24e^5-40e^4+4e^3+8e^2+6e+3}{6e^5-9e^4-e^3+2e^2+3e+3}$ |
| 4 | $2\frac{50e^7-133e^6+92e^5+7e^4-7e^3-6e^2-4e-2}{20e^7-50e^6+28e^5+8e^4-3e^2-4e-4}$ |

for $k = 2$ is

$$S_D(C|e) = \frac{18e^5 - 8e^4 - 5e^3 - 6e^2 - 4e - 2}{4e^5 - e^4 - e^3 - 2e^2 - 2e - 2},$$

and for $k = 3$ is

$$S_D(C|e) = \frac{126e^8 - 183e^7 + 23e^6 - 2e^5 + 16e^4 + 10e^3 + 9e^2 + 6e + 3}{21e^8 - 27e^7 + e^6 - 3e^5 + 2e^4 + 2e^3 + 3e^2 + 3e + 3}.$$

# 7 Simulation Results

This section covers the setup of the simulations and presents the results. In order for the theoretical and simulated results to be comparable, all erasures are i.i.d. and no retransmissions are made. We compare our simulation results with the theoretical analysis of the symbol loss probability and the symbol delay for both the single hop and recoding networks. A Monte Carlo simulation of the two networks is conducted for different values of $k$. For each $k$ the simulation is run with the erasure probabilities $e \in \{0.05, 0.10, \ldots, 0.95\}$. Each $(k, e)$-pair is simulated with $10 \times 10^6$ repetitions. The region of interest is defined as $0 \leq e \leq 1 - r$, where $r$ is the rate of the code.

**Table B.4:** The largest absolute differences (percentage points) using the single hop network. Gray indicate RLNC.

| Rate | Type | Theoretical | Superregular |
|---|---|---|---|
| 1/2 | Simulation | $0.029, k = 6, e = 0.60$ | |
| | | $0.102, k = 1, e = 0.70$ | $0.098, k = 1, e = 0.50$ |
| | Experiments | $1.467, k = 6, e = 0.31$ | |
| | | $1.412, k = 6, e = 0.31$ | $0.055, k = 6, e = 0.31$ |
| 1/3 | Simulation | $0.023, k = 3, e = 0.75$ | |
| | | $0.117, k = 1, e = 0.70$ | $0.116, k = 1, e = 0.65$ |
| | Experiments | $0.511, k = 4, e = 0.31$ | |
| | | $0.490, k = 4, e = 0.31$ | $0.021, k = 4, e = 0.31$ |

## 7.1 Random Linear Network Coding

We focus on the field GF($2^8$), since its elements fit perfectly into a single byte. Thus, a field of this size eases the implementation of Galois field arithmetic and facilitates a high efficiency. Specifically, we use the primitive polynomial

$$f_p(\omega) = \omega^8 + \omega^4 + \omega^3 + \omega^2 + 1,$$

where

$$\omega \in \Omega_{f_p} = \{2, 4, 16, 29, 76, 95, 133, 157\}.$$

In this section, we compare the performance of the superregular lower triangular Toeplitz matrices with the performance of lower triangular RLNC. A lower triangular RLNC code is generated by choosing the coding coefficients at random, using an uniform distribution on all elements of the field [14]. That is, using the field GF($2^8$) a coding coefficient is set to zero with probability $2^{-8}$. Another approach could be to exclude the zero element of the field when generating coding coefficients. This would produce slightly different results, and the effect is not covered in this paper.

## 7.2 Symbol Loss Probability

For the single hop network the simulation is conducted with $k \in \{1, \ldots, 10\}$ for codes with rate 1/2 and with $k \in \{1, \ldots, 7\}$ for codes with rate 1/3. In Fig. B.4a, the simulated results when using a superregular lower triangular Toeplitz matrix and RLNC is shown with $\circ$ and $+$ respectively. Fig. B.4b shows the symbol loss probability for the case of rate 1/3 codes. The largest absolute differences between the theoretical and simulation results (for both superregular matrices and RLNC) are listed in Table B.4. In all cases the superregular matrices are favorable over RLNC.

Simulations of the recoding network use codes with $k \in \{1, \ldots, 6\}$. In Fig. B.5a (equal erasure probability) and B.5b (unequal erasure probability)

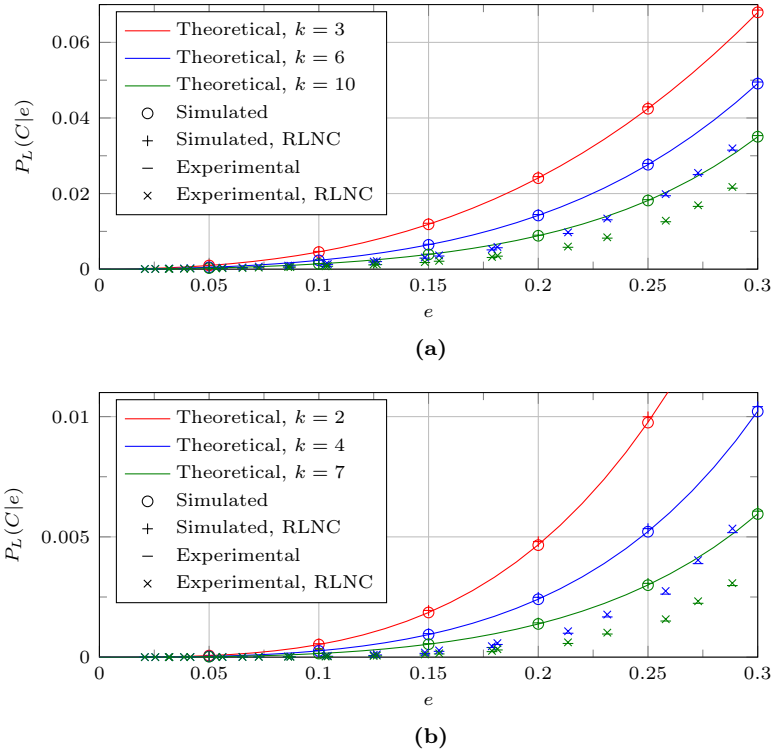
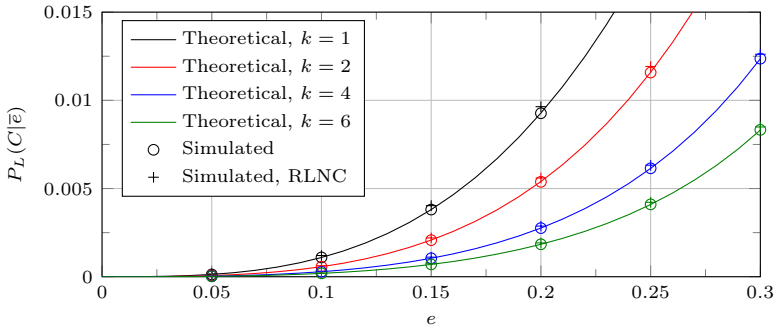



**Fig. B.4:** Symbol loss probability for the single hop network when using: (a) superregular lower triangular Toeplitz matrices ($n = 2k$), (b) two jointly superregular lower triangular Toeplitz matrices ($n = 3k$).

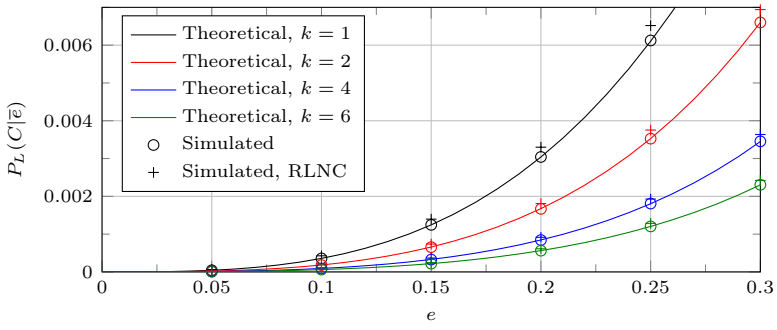
the simulated results when using product preserving jointly superregular lower triangular Toeplitz matrices and RLNC are shown with $\circ$ and $+$ respectively. For the case of unequal erasure probability, the product preserving jointly superregular lower triangular matrices are favorable for all the tested values of $k$ and $e$. The largest absolute differences between the theoretical and simulation results (for both product preserving jointly superregular matrices and RLNC) are listed in Table B.5. For each of the absolute differences the superregular matrices are the favorable coding matrices.

**Table B.5:** The largest absolute differences (percentage points) using the recoding network. Gray indicate RLNC.

| $e$ | **Theoretical** | **Superregular** |
|---|---|---|
| Equal | $0.015, k = 6, e = 0.65$ | |
| | $0.125, k = 1, e = 0.55$ | $0.128, k = 1, e = 0.55$ |
| Unequal | $0.013, k = 2, e = 0.80$ | |
| | $0.738, k = 6, e = 0.90$ | $0.733, k = 6, e = 0.90$ |




**(a)**




**(b)**

**Fig. B.5:** Symbol loss probability for the recoding network when using product preserving jointly superregular lower triangular Toeplitz matrices and RLNC. (a) equal erasure probability, (b) unequal erasure probability.

## 7.3 Symbol Delay

For the single hop network the symbol delay simulations are conducted with $k \in \{1, \ldots, 10\}$ for codes with rate $1/2$ and with $k \in \{1, \ldots, 7\}$ for codes with rate $1/3$. Fig. B.7 compares our simulation results with the theoretical analysis of the symbol delay. Fig. B.7a compares the rate $1/2$ codes and Fig. B.7b compares the rate $1/3$ codes. From the figures it is evident that the theoretical and simulated symbol delay coincide. The codes based on superregular matrices outperform RLNC in all tested scenarios where $k > 1$.

There exists a trade-off between symbol loss probability and delay. For example, with a systematic code the symbol delay can be minimized by adjusting the code parameters such that the symbol loss probability is increased. Thus, a code must be designed such that the symbol loss probability and delay match the specifications of the application in question. Fig. B.6 shows the results (in terms of symbol loss probability and delay) of simulating the single hop network with two different codes, namely RLNC and a code based on a superregular lower triangular matrix for $k = 1$. Both codes operate over GF(2). It can be observed that RLNC provides a slightly lower symbol delay, while the symbol loss probability is significantly higher.

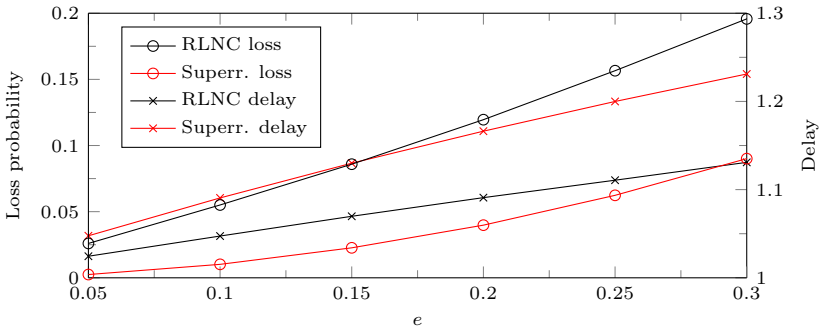



**Fig. B.6:** Simulated symbol loss probability and delay for the single hop network when using (B.6) and RLNC over GF(2) for $k = 1$.

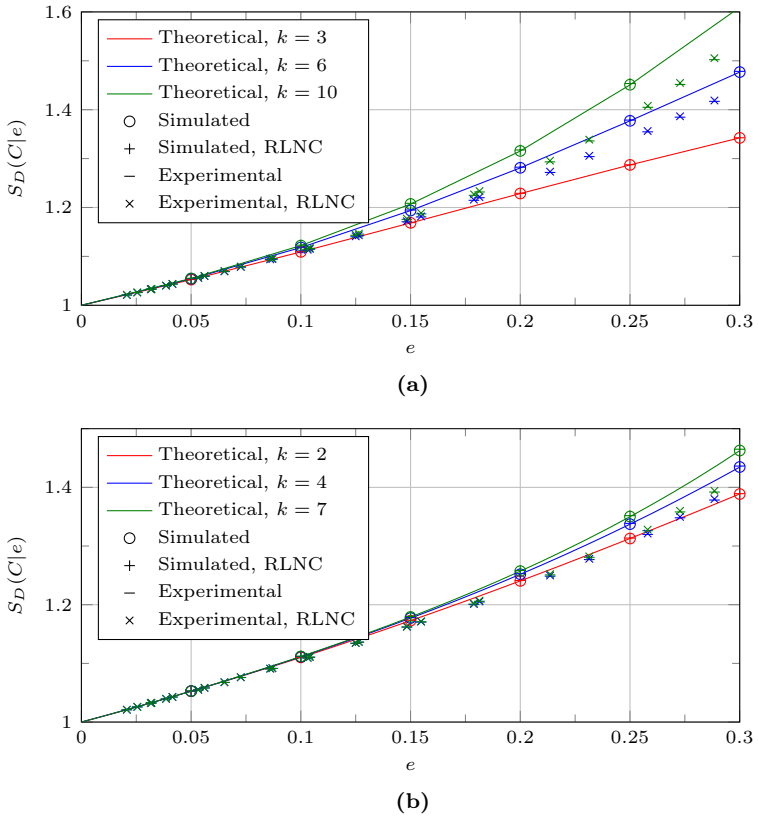



**Fig. B.7:** Symbol delay for the single hop network when using: (a) superregular lower triangular Toeplitz matrices ($n = 2k$), (b) two jointly superregular lower triangular Toeplitz matrices ($n = 3k$).

# 8 Experimental Results

In our experiments we use the same definition of RLNC as the one used in the simulations, which is stated in Section 7.1. In order to produce comparable results between both different values of $k$ and the two coding methods, superregular matrices and RLNC, a number of channel models have been obtained. The models are based on the N-state Markov model proposed by [27], and we refer to this as the extended Gilbert model (EGM). The channel models are obtained using a testbed consisting of two nodes (one transmitting and one receiving) described in Table B.6. The nodes are connected using wires, instead of using the normal antennas. This design seeks to eliminate external noise sources and to ensure that the testbed produces consistent and reproducible results. The nodes communicate using IEEE 802.11g in IBSS mode,

despite the wired connections. The transmitting node broadcasts packets using the highest data rate (54 Mbit/s). This ensures maximum capacity and disables any retransmissions, since broadcast packets are never retransmitted. Both the experiments and simulations could have included things like MAC-layer retransmissions, link-layer coding or even packet aggregation. However, these optional Wi-Fi enhancements were considered application specific and not concise, e.g., how many retransmissions should be used? Which type of code? and how many packets to aggregate? The erasures of the wireless channel are introduced by attenuating the signal, and thus decreasing the signal–to–noise ratio (SNR). The obtained channel models produce average erasure probabilities between 0.0207 and 0.9910, with a corresponding average burst length of 1.0110 and 110.6940 packets.

## 8.1 Single Hop Network

Simulations of the single hop network are conducted where the quality of the link between the nodes is based on the obtained EGM channel models. In Fig. B.4a the experimental symbol loss probability (for rate $1/2$ codes) when using a superregular lower triangular Toeplitz matrix and RLNC are shown with $-$ and $\times$, respectively. Observing Fig. B.4a it becomes apparent that the experimental results follow a trend similar to that of the theoretical and simulated results. Fig. B.4b shows the symbol loss probability for codes with a rate of $1/3$. The largest absolute differences between the theoretical and experimental results (for both superregular matrices and RLNC) are listed in Table B.4. The difference between the two methods is almost negligible, although the superregular matrices produce a slightly lower symbol loss probability.

**Table B.6:** Details of the wireless nodes used in the experiments.

| **Type** | **Name** |
|---|---|
| CPU | Intel Core i3-4170 |
| OS | Ubuntu Linux 15.10 |
| Kernel | Linux 4.4.11 |
| WNIC | Compex WLE600VX |
| WNIC driver | ath10k |
| WNIC firmware | 10.1.467 api 2 |

## 8.2 Recoding Network

Our experiments with recoding at an intermediate node use the logical recoding network shown in Fig. B.3b. In Fig. B.8 the result of three sets of experiments are shown. All of the experiments are with $k = 4$. In the experiments, the erasures on the links are modeled using the obtained EGM channel models. For all three experiments, the erasure rate on the link between A and R is fixed to $e_1 = 0.29$ with an average burst length of 1.2941 packets. The direct link, between A and B, is set to simulate three different EGM channel models. The three models have the following erasure rates: 0.23, 0.81, and 0.91, with corresponding average burst lengths of: 1.1899, 5.3234, and 14.4615. For the last link all obtained EGM channel models are used. It is clear that the use of a more advanced channel model shows different results, compared to a channel with i.i.d. losses. Regardless of the channel model the use of product preserving jointly superregular lower triangular Toeplitz matrices yields a lower symbol loss probability compared to RLNC.

## 8.3 Coding Throughput

In [19] we showed that the combined encoding and decoding throughput is significantly increased by using superregular lower triangular Toeplitz matrices with several ones in the first column of the matrix. This gain is also valid for (product preserving jointly) superregular lower triangular Toeplitz matrices when compared to RLNC. Fig. B.9 shows the combined gain in encoding and decoding throughput when using the matrices in (B.6) and (B.7) over lower triangular RLNC. Additionally, Fig. B.9 also shows the gain when recoding is performed using the matrices in (B.10) for encoding and (B.11) for recoding over lower triangular RLNC with recoding.

The test procedure without recoding is as follows. A vector with 1500 elements is encoded using a single row of a superregular lower triangular Toeplitz or random matrix. The coded vector is fed into the decoder, which then performs Gaussian elimination. These two steps are performed until the decoder is able to decode all $k$ symbols. To ensure reliable results, the experiment is repeated $200 \times 10^6$ times, for each $k$. When all repetitions are completed, the mean throughput is calculated. The throughput gain is calculated as the ratio between the throughput when using a superregular matrix and the throughput when using a random matrix. When recoding is included, the encoder feeds the coded vector to a recoder which performs decoding (Gaussian elimination) and recoding before feeding the new coded vector to the decoder, which also performs Gaussian elimination. The experiment was carried out using a PC described in Table B.6.

The gain without recoding is, as expected, identical for the two superregular lower triangular Toeplitz matrices for $k = 1$ and $k = 2$. For these values of $k$

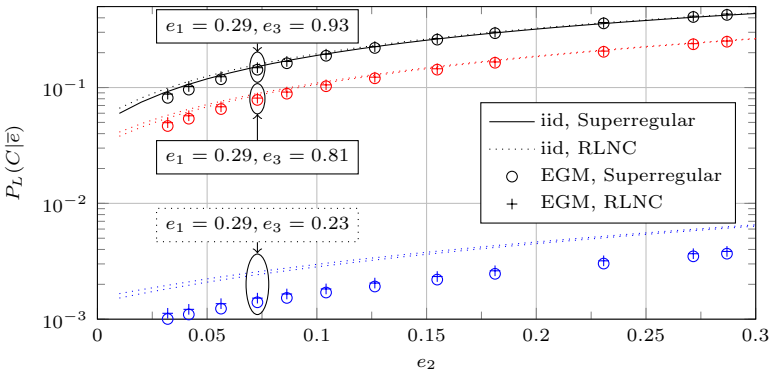



**Fig. B.8:** Simulated symbol loss probability when using product preserving jointly superregular lower triangular Toeplitz matrices and RLNC with $k = 4$ in the recoding network, where the erasures are i.i.d. and based on the EGM channel models.

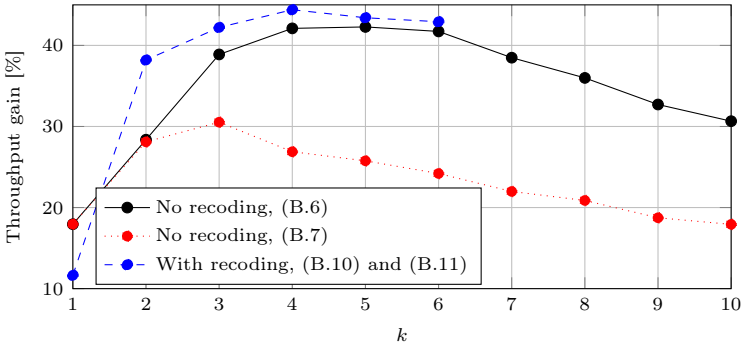



**Fig. B.9:** Gain in coding throughput of (product preserving jointly) superregular lower triangular Toeplitz matrices over lower triangular RLNC.

the two matrices require the same number of multiplications during encoding and decoding. For $k > 2$ the gain using (B.6) is larger than using (B.7), as there are fewer multiplications during encoding and decoding. Eq. (B.6) has ones in at least one and in at most three diagonals (depending on $k$), whereas (B.7) only has ones in the main diagonal. For $k = 5$ the gains are 42.3 % and 25.8 % using (B.6) and (B.7), respectively.

The gain when recoding is performed is negligible for $k = 1$, but for $k > 1$ the gain is considerable. This is again due to a reduced number of multiplications. Eq. (B.10) has ones on the first two diagonals, and again on the fifth diagonal, whereas, (B.11) has ones on the first and third diagonals. For $k = 4$ the gain is 44.4 % over a random based code.

**Table B.7:** Number of operations for encoding and decoding. For $k = 5$ and a $5 \times 1500$ source data matrix

| **State** | **Operation** | **RLNC** | **Eq.** (B.6) | **Eq.** (B.7) |
|---|---|---|---|---|
| Encoding | Addition | 15058.3 | 15000 | 15000 |
| | Multiplication | 22469.5 | 7500 | 15000 |
| Decoding forward | Subtraction | 15049.5 | 15050 | 15050 |
| | Multiplication | 22486.1 | 15050 | 21070 |
| | Inversion | 14.9 | 10 | 14 |
| Decoding backward | Subtraction | 58.6 | 0 | 0 |
| | Multiplication | 116.6 | 0 | 0 |
| | Inversion | 0.1 | 0 | 0 |

## 8.4 Coding Operations

As stated earlier, the gain in throughput comes mainly from fewer multiplications and partly from fewer inversions during encoding and decoding. Table B.7 lists the number of operations needed when encoding and decoding using (B.6), (B.7) and RLNC, for $k = 5$ and a $5 \times 1500$ source data matrix. For RLNC, the number of operations is calculated by counting and averaging the number of operations used in the simulations described in the previous section. Furthermore, for RLNC both forward and backward substitution are required when decoding, due to the possibility of linear dependency of the rows. It is, however, clear from Table B.7 that the computational effort lies in the forward substitution part of the decoding process.

### Encoding

The number of additions during encoding for the three methods are comparable. However, the three methods require significantly different numbers of multiplications. For RLNC all matrix elements are larger than one, with probability $^{253}/_{254}$, thus the number of multiplications is approximately $15 \cdot 1500 = 22500$. The probability that elements are either one or zero slightly reduces the number of operations. On the other hand, the number of operations is increased by the probability of linear dependency, and the subsequent cost of encoding additional rows.

Both superregular lower triangular Toeplitz matrices each have 15 nonzero matrix elements. The matrix defined in (B.7) only has ones on the diagonal, so $(15 - 5) \cdot 1500 = 15000$ multiplications have to be performed. Using the other superregular matrix, which in total has 10 ones, requires just $(15 - 10) \cdot 1500 = 7500$ multiplications.

### Decoding

When decoding, Gaussian elimination must be performed on both the coded data and the coding matrix. This is equivalent to concatenating the two matrices and putting the coding matrix part in reduced row echelon form. That is, during decoding, each row has $5 + 1500 = 1505$ elements. This decoding procedure is preferred over multiplying by the inverse of the coding matrix for two reasons. First of all, in a practical implementation, Gaussian elimination is used when a packet is received from the network to reduce the latency, as opposed to waiting for a full rank matrix and then inverting and decoding. Secondly, this procedure eliminates the potential need for using the pseudo inverse, when the number of received code words is larger than $k$.

The number of subtractions for the three methods are, like the number of additions during encoding, quite similar. However, the number of multiplications are very different. For RLNC the number of multiplications during encoding and decoding are nearly the same, simply because the matrix elements have a high probability of being larger than one. By the same token, the number of inversions is nearly equal to the number of elements. Backward substitution requires little to no effort, compared to forward substitution.

The number of ones in the two superregular lower triangular Toeplitz matrices also has an effect on the decoding process. Eq. (B.6) needs $10 \cdot 1505 = 15050$ multiplications, which also corresponds to the number of inversions, 10, whereas (B.7) needs $14 \cdot 1505 = 21070$ multiplications, which then corresponds to the number of inversions, 14.

## 9 Discussion

It is clear that the theoretical and simulated (with i.i.d. erasures) symbol loss probabilities and symbol delays coincide, when using (product preserving jointly) superregular lower triangular Toeplitz matrices. It is also clear from the simulations with i.i.d. erasures that the difference in symbol loss probabilities and symbol delays between (product preserving jointly) superregular lower triangular Toeplitz matrices and RLNC are, for the most part, negligible. As expected, erasures are not i.i.d. on a real channel, so the simulated (with i.i.d. erasures) and theoretical symbol loss probabilities and symbol delays do not coincide with the simulated (with EGM channel models) symbol loss probabilities and symbol delays. However, they do follow similar trends. Finally, simulated (with EGM channel models) symbol loss probabilities and symbol delays for superregular lower triangular Toeplitz codes are comparable to those of RLNC, but it is also visible that there is a small gain by using (product preserving jointly) superregular lower triangular Toeplitz matrices over RLNC.

# 10 Conclusions

In this paper we have presented both explicit matrix constructions and a greedy search algorithm for superregular lower triangular Toeplitz matrices. We have defined two matrix attributes for lower triangular matrices, namely *joint superregularity* and *product preserving joint superregularity*, and we extended our explicit matrix constructions to include these two cases. We motivated the use of (product preserving) jointly superregular matrices, in general, with use-cases such as intermediate recoding and codes with a rate greater than or equal to $1/3$. In both cases, optimal decoding capabilities are of great interest. Furthermore, we also showed several general attributes of (jointly) superregular matrices

We provided a general method for deriving the exact symbol loss probability when encoding (and recoding) with (product preserving) jointly superregular lower triangular Toeplitz matrices using an erasure channel with i.i.d. erasures. We also provided the exact symbol loss probability for a large number of different scenarios, code rates, and values of $k$.

We then extended our theoretical analysis to the exact symbol delay for any static code using an erasure channel with i.i.d. erasures. Moreover, we used the single hop network to exemplify the exact symbol delay for two different code rates and different values of $k$.

In all simulated scenarios, (product preserving jointly) superregular lower triangular Toeplitz coding matrices were found to be favorable compared to RLNC for all the tested values of $k$ (except for the symbol delay when $k = 1$) within the region of interest. That is, all three types of block codes using superregular matrices produced a lower average symbol loss probability than RLNC, with or without recoding and regardless of the rate.

We also showed that the encoding and decoding performance is greatly improved by using superregular lower triangular Toeplitz matrices with several ones. The coding throughput is increased by 42.3 % without recoding ($k = 5$) and by 44.4 % with recoding ($k = 4$), simply by reducing the number of multiplications and to some degree the number of inversions.

# A Proof of Lemma B.3

i) Since $\omega^{i_1} \neq 0$ then $A_2$ is always superregular.

ii) The determinants of the proper submatrices of $A_3$ are: $\omega^{i_1}$ and $\omega^{2i_1} \oplus \omega^{i_2}$, where $\omega^{i_1} \neq 0$. Since $\omega$ is primitive, $\omega^i \neq \omega^j$, if $(i, j) \in \mathcal{I}_2$, $i \neq j$. Thus, $\omega^{2i_1} \oplus \omega^{i_2} \neq 0 \Leftrightarrow 2i_1 \neq i_2$ (modulo $2^p - 1$).

iii) We only need to check the determinants of the proper submatrices that include the new element $\omega^{i_3}$. Since $\omega$ is primitive, it is easy to obtain (B.1).

iv) It is easy to obtain the determinant expressions of the proper submatrices that include the new element $\omega^{i_4}$. These expressions contain terms in the form $\omega^i \oplus \omega^i$. Since arithmetic operations are wrt. GF($2^p$), $\omega^i \oplus \omega^i = 0, \forall\omega, \forall i$, and we obtain (B.2) and (B.3). □

# References


[1] S. A. Hosseini, Z. Lu, G. de Veciana, and S. S. Panwar, "SVC-Based Multi-User Streamloading for Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 8, pp. 2185–2197, August 2016.

[2] S. Hahm, P. Kang, H. Bang, and H. Yeon, "Dynamic media buffer control scheme for seamless streaming in wireless local area networks," *IEEE Wireless Communications and Networking Conference Workshops*, pp. 109–114, 2016.

[3] M. Muntner and J. Wolf, "Predicted performance of error-control techniques over real channels," *IEEE Transactions on Information Theory*, vol. 14, no. 5, pp. 640–650, September 1968.

[4] X. Chen, N. Ren, X. Zhang, X. Wang, and J. Zhao, "SonicStream: A network coding based live P2P media streaming system with rich user experiences," *Journal of Communications and Networks*, vol. 10, no. 4, pp. 430–436, December 2008.

[5] J. H. Sørensen, P. Popovski, and J. Østergaard, "Delay Minimization in Real-Time Communications With Joint Buffering and Coding," *IEEE Communications Letters*, vol. 21, no. 1, pp. 52–55, January 2017.

[6] A. Majumda, D. G. Sachs, I. V. Kozintsev, K. Ramchandran, and M. M. Yeung, "Multicast and unicast real-time video streaming over wireless LANs," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 6, pp. 524–534, 2002.

[7] P. Elias, "Coding for noisy channels," *IRE Convention Records Part 4*, pp. 37–46, 1955.

[8] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, April 1950.

[9] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[10] J. K. Sundararajan, D. Shah, M. Médard, S. Jakubczak, M. Mitzenmacher, and J. Barros, "Network Coding Meets TCP: Theory and Implementation," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 490–512, 2011.

[11] J. Hansen, J. Krigslund, D. E. Lucani, and F. H. P. Fitzek, "Sub-Transport Layer Coding: A Simple Network Coding Shim for IP Traffic," *IEEE 80th Vehicular Technology Conference*, pp. 1–5, 2014.

[12] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the Air: Practical wireless network coding," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 497–510, June 2008.

[13] P. Pahlevani, D. Lucani, M. Pedersen, and F. Fitzek, "PlayNCool: Opportunistic network coding for local optimization of routing in wireless mesh networks," *IEEE Globecom Workshops*, pp. 812–817, December 2013.

[14] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," *IEEE International Symposium on Information Theory*, June 2003.

[15] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 169–180, August 2007.

[16] J. Krigslund, J. Hansen, M. Hundebøll, D. Lucani, and F. Fitzek, "CORE: COPE with MORE in Wireless Meshed Networks," *IEEE 77th Vehicular Technology Conference*, June 2013.

[17] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, October 2006.

[18] J. Heide, M. Pedersen, F. Fitzek, and M. Médard, "On Code Parameters and Coding Vector Representation for Practical RLNC," *IEEE International Conference on Communications*, pp. 1–5, June 2011.

[19] J. Hansen, J. Østergaard, J. Kudahl, and J. H. Madsen, "On the Construction of Jointly Superregular Lower Triangular Toeplitz Matrices," *IEEE International Symposium on Information Theory*, pp. 1–5, July 2016.

[20] A. Paramanathan, P. Pahlevani, S. Thorsteinsson, M. Hundebøll, D. E. Lucani, and F. H. P. Fitzek, "Sharing the Pi: Testbed Description and Performance Evaluation of Network Coding on the Raspberry Pi," *IEEE 79th Vehicular Technology Conference*, 2014.

[21] R. Roth and A. Lempel, "On MDS codes via Cauchy matrices," *IEEE Transactions on Information Theory*, vol. 35, no. 6, pp. 1314–1319, November 1989.

[22] H. Gluesing-Luerssen, J. Rosenthal, and R. Smarandache, "Strongly-MDS convolutional codes," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 584–598, February 2006.

[23] M. Aissen, I. Schoenberg, and A. Whitney, "On the generating functions of totally positive sequences I," *Journal d'Analyse Mathématique*, vol. 2, no. 1, pp. 93–103, 1952.

[24] P. Almeida, D. Napp, and R. Pinto, "A new class of superregular matrices and MDP convolutional codes," *Linear Algebra and its Applications*, vol. 439, no. 7, pp. 2145–2157, 2013.

[25] R. Smarandache, H. Gluesing-Luerssen, and J. Rosenthal, "Strongly MDS convolutional codes, a new class of codes with maximal decoding capability," *IEEE International Symposium on Information Theory*, January 2002.

[26] R. Mahmood, A. Badr, and A. Khisti, "Convolutional codes with maximum column sum rank for network streaming," *IEEE International Symposium on Information Theory*, pp. 2271–2275, June 2015.

[27] V. K. Varma, "Testing speech coders for usage in wireless communications systems," *IEEE Workshop on Speech Coding for Telecommunications*, pp. 93–94, 1993.

# Paper C

## On Superregular Matrices and Convolutional Codes with Finite Decoder Memory

Jonas Hansen, Jan Østergaard, Johnny Kudahl, and John H. Madsen

The paper has been submitted to the
*IEEE 87th Vehicular Technology Conference*, 2018.

# Abstract

*In this paper, we present explicit code constructions for a family of $(n, k, \delta)$ convolutional codes with optimum distance profiles. The family of convolutional codes is obtained from sets of jointly superregular matrices. For the case of finite decoder memory, we evaluate the performance of the constructed codes in terms of both symbol loss probability and symbol delay. We then present a combinatorial method to calculate the exact symbol loss probability and symbol delay for each symbol individually. We compare the symbol loss probability for two specific systematic convolutional codes for the cases where the sink has infinite or finite memory. Finally, we compare the performance of our convolutional codes with optimum distance profile and random based convolutional codes.*


# 1 Introduction

It was shown in [1] that any dense superregular matrix can be used to construct a systematic maximum distance separable (MDS) block code. A lower triangular matrix is considered superregular, if and only if all of its proper submatrices are non-singular [2]. The authors of [2] showed that an MDS convolutional code can be constructed from superregular lower triangular Toeplitz matrices. In [3], a new class of lower block triangular matrices that are superregular over a sufficiently large field was presented. An upper bound on the minimum finite field size such that a superregular matrix of a given size can exist over that field was shown in [4]. In [5], the authors presented constructions of convolutional codes that attain the maximum possible distance for some fixed parameters of the code, that is, the rate and the Forney indices. One of the benefits of convolutional codes [6] is their remarkable ability to recover from burst losses [7], however, this ability of course depends on the parameters of the code. Hence, convolutional codes are appropriate for streaming, given the bursty nature of modern wireless networks.

In this work we provide explicit code constructions for $(n, k, \delta)$ convolutional codes with memory $v$ and optimum distance profile (ODP) [8]. Specifically, we show that these codes can be constructed from sets of jointly superregular matrices. We show that a set of jointly superregular matrices can be obtained using an algorithm similar to the one showed in [9]. Finally, we present a method to calculate the symbol loss probability and symbol delay, and use this to show the effect of using finite decoder memory.

## 2 Jointly Superregular Matrices

Let $\Omega_{f_p}$ denote the set of roots of a primitive polynomial $f_p$, which generates $\mathbb{F}_{2^p}$. Let $\mathcal{I}_k \triangleq \{i_1, \ldots, i_k : i_j \in \mathbb{F}_{2^p},\ i_j \neq 2^p - 1,\ \forall j\}$. Let $\Psi(i_1, \ldots, i_k)$ be a $k \times k$ lower triangular Toeplitz matrix with the first column given by $[i_1, \ldots, i_k]^T$. Let $\psi_\omega(i_1, \ldots, i_{k-1}) = \Psi(1, \omega^{i_1}, \ldots, \omega^{i_{k-1}})$ be a $k \times k$ lower triangular Toeplitz matrix with ones on the main diagonal and all non-zero elements below the diagonal. Let $\omega \in \Omega_{f_p}$ and let $\mathcal{A}_k^\omega$ denote the set of all $k \times k$ superregular lower triangular Toeplitz matrices given by $\psi_\omega(i_1, \ldots, i_{k-1})$, where $(i_1, \ldots, i_{k-1}) \in \mathcal{I}_{k-1}$.

In [1], the authors defined a matrix to be superregular if and only if every square submatrix is non-singular. A lower triangular matrix is defined in [2, Definition 3.3] to be superregular if and only if every proper submatrix is non-singular. The definition of a proper submatrix is as follows. Let $A$ be a $k \times k$ lower triangular matrix. Let $A' = A_{h_1,\ldots,h_r}^{j_1,\ldots,j_r}$ be an $r \times r$ submatrix of $A$, where $A'$ is constructed using the rows and columns of $A$ with indices $j_1, \ldots, j_r$ and $h_1, \ldots, h_r$, respectively [2, Definition 3.2]. Then, $A'$ is a proper submatrix of $A$ if and only if $1 \leq j_1 < j_2 < \ldots < j_r \leq k$, $1 \leq h_1 < h_2 < \ldots < h_r \leq k$ and $j_t \geq h_t, \forall t$.[1]

The definition of joint superregularity for the case of two lower triangular Toeplitz matrices was given in [9]. In Definition C.1 we extend this definition to multiple matrices.

**Definition C.1 (Joint superregularity).**
*A set of $M$, where $M \in \{2, 3, \ldots\}$, superregular lower triangular Toeplitz $k \times k$ matrices are jointly superregular if and only if all of the proper submatrices of the concatenated $Mk \times k$ matrix are non-singular. In this context a proper submatrix is any square matrix that is not trivially rank deficient. An $t \times t$ matrix, when sorted by increasing row support size[2], is said to be trivially rank deficient if the support of row $i$, $1 \leq i \leq t$, is less than $i$.* △

A greedy algorithm for obtaining $M$ jointly superregular lower triangular Toeplitz matrices with dimensions $k \times k$ can be constructed based on [9, Algorithm 1]. That is, at each instance in the algorithm where all of the M matrices are superregular, they are tested for joint superregularity using Def. C.1. If this is the case and the matrices have dimensions $k \times k$ the search stops. If the dimensions are not $k \times k$ then both matrix dimensions are increased by one and the search continues. If the matrices are not jointly superregular the algorithm continues to search for superregular matrices of the same dimensions. Let $\mathcal{B}_{k,M}^\omega$ denote the set of all sets of $M$ different $k \times k$ jointly superregular lower triangular Toeplitz matrices according to Definition C.1.

[1] In [10] the authors use a slightly different notion of superregularity.
[2] The size of the support of a vector is equal to the number of its non-zero elements.

# 3 Convolutional codes

Let $\mathcal{C} \subseteq \mathbb{F}[D]_{2^p}^n$ be an $(n, k, \delta)$ convolutional code with a basic minimal generator matrix

$$G = \sum_{j=0}^{v} G_j D^j \in \mathbb{F}[D]_{2^p}^{k\times n}, G_j \in \mathbb{F}_{2^p}^{k\times n}, G_v \neq 0,$$

where $v$ is the memory of the code. We define the degree of $\mathcal{C}$, $\delta$, as the sum of the row degrees of $G$ [2]. Denoted by $G_j^c, j \in \mathbb{N}_0$ the truncated sliding generator matrix

$$G_j^c = \begin{bmatrix} G_0 & G_1 & \cdots & G_j \\ & G_0 & \cdots & G_{j-1} \\ & & \ddots & \vdots \\ & & & G_0 \end{bmatrix},$$

where $G_j = 0$ whenever $j > v$. The reader is referred to [8] for further details. In [2, Proposition 2.2] an upper bound on the column distance is given, that is, for every $j \in \mathbb{N}_0$ then

$$d_j^c \leq (n-k)(j+1)+1, \tag{C.1}$$

where $d_j^c$ is the $j$th column distance. In [8] the authors define the distance profile of the code $\mathcal{C}$ to be $\mathbf{d}^p = (d_0^c, d_1^c, \ldots, d_v^c)$. A convolutional code is said to have ODP [8] if and only if the $v+1$ elements of $\mathbf{d}^p$ attain the bound given in (C.1). From [2, Corollary 2.3] we have that, if $d_j^c$ for some $j \in \mathbb{N}_0$ attains the bound in (C.1) then $d_i^c$ for $i \leq j$ also attain the bound in (C.1). Thus, a convolutional code has ODP if and only if $d_v^c$ attains the bound in (C.1). Finally, if a convolutional code has ODP, it is said to be an ODP convolutional code.

**Lemma C.1.** *Let $\Psi(i_0^{(l')}, \ldots, i_v^{(l')}) \in \mathbb{F}_{2^p}^{(v+1)\times(v+1)}$, $l' = 1, \ldots, l$, be a set of $l$ lower triangular Toeplitz matrices, where $i_v^{(l')} \neq 0$. Moreover, let $nk = l$. Then, an $(n, k, \delta)$ convolutional code with memory $v$, $\delta = vk$, and basic minimal generator matrix $G$ can be constructed as*

$$G = \begin{bmatrix} \sum_{j=0}^{v} i_j^{(1)} D^j, \ldots, \sum_{j=0}^{v} i_j^{(n)} D^j \\ \vdots \qquad\qquad \ddots \qquad\qquad \vdots \\ \sum_{j=0}^{v} i_j^{(l-n+1)} D^j, \ldots, \sum_{j=0}^{v} i_j^{(l)} D^j \end{bmatrix}. \tag{C.2}$$

△

Fig. C.1 shows how to construct $G_v^c$, for a $(2, 1, 3)$ convolutional code with $v = 3$, from two $4 \times 4$ lower triangular Toeplitz matrices. It is trivial

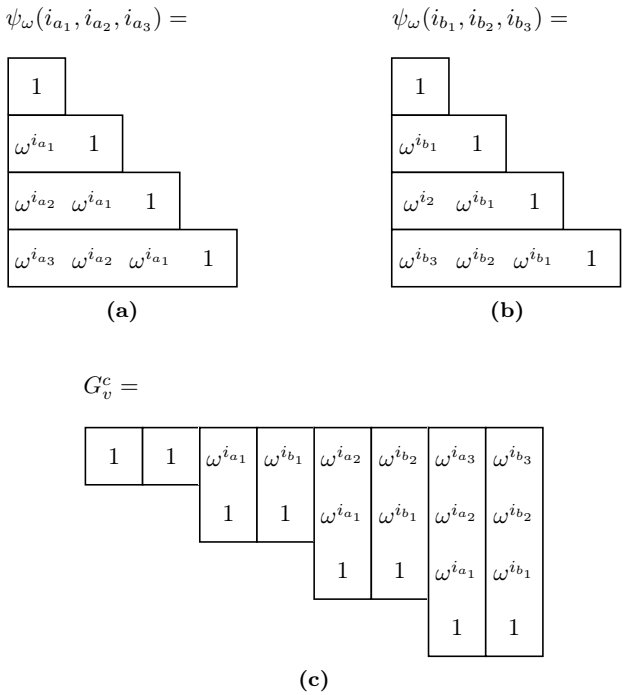



**Fig. C.1:** A $(2, 1, 3)$ convolutional code with $v = 3$ constructed from the two matrices in (a) and (b) has $G_v^c$ as shown in (c).

to construct $G$ from $G_v^c$, thus the actual proof is omitted due to space considerations.

**Lemma C.2.** *If the $l$ matrices of Lemma C.1 are jointly superregular, then the $(n, k, \delta)$ convolutional code with memory $v$ and basic minimal generator matrix $G$ in* (C.2) *is an ODP convolutional code.* △

*Proof.* In [2, Theorem 2.4] the authors state that the following are equivalent:

i) $d_j^c = (n - k)(j + 1) + 1$;

ii) every $(j + 1)k \times (j + 1)k$ full-size minor of $G_j^c$ formed from the columns with indices $1 \leq t_1 < \cdots < t_{(j+1)k}$, where $t_{sk+1} > sn$ for $s = 1, \ldots, j$, is nonzero.

If i) is true for all $j \in \{0, \ldots, v\}$ then the code is an ODP convolutional code, and ii) is for $j \in \{0, \ldots, v\}$ guaranteed by the definition of joint superregularity (see Definition C.1). This can be realised by concatenating (as shown in Fig. C.1) the $nk$ jointly superregular lower triangular Toeplitz matrices and the resulting matrix is then identical to $G_v^c$. □

We now provide two examples of sets of jointly superregular lower triangular Toeplitz matrices which we obtained using the previously described method. We then use Lemma C.1 to construct $(n, k, \delta)$ ODP convolutional codes.

**Example C.1.** *A $(2, 1, 5)$ ODP convolutional code over $\mathbb{F}_{2^8}$ may be constructed using the two jointly superregular lower triangular Toeplitz matrices*

$$A_6^{(1)} = \psi_\omega(0, 2, 5, 0, 15),$$
$$A_6^{(2)} = \psi_\omega(1, 0, 4, 9, 30),$$

*that is $(A_6^{(1)}, A_6^{(2)}) \in \mathcal{B}_{6,2}^{\omega}$. The basic minimal generator matrix is then*

$$G = [1 + D + \omega^2 D^2 + \omega^5 D^3 + D^4 + \omega^{15} D^5,$$
$$1 + \omega D + D^2 + \omega^4 D^3 + \omega^9 D^4 + \omega^{30} D^5].$$

*The column distances are: $d_j^c = j + 2$ for $0 \leq j \leq 5$.*

**Example C.2.** *A $(3, 2, 8)$ ODP convolutional code over $\mathbb{F}_{2^8}$ may be constructed using the six jointly superregular lower triangular Toeplitz matrices*

$$B_5^{(1)} = \psi_\omega(50, 12, 147, 114),$$
$$B_5^{(2)} = \psi_\omega(181, 79, 29, 67),$$
$$B_5^{(3)} = \psi_\omega(112, 235, 226, 191),$$
$$B_5^{(4)} = \psi_\omega(103, 155, 45, 0),$$
$$B_5^{(5)} = \psi_\omega(67, 0, 160, 88),$$
$$B_5^{(6)} = \psi_\omega(55, 177, 183, 161),$$

*that is $(B_5^{(1)}, \ldots, B_5^{(6)}) \in \mathcal{B}_{5,6}^{\omega}$. The basic minimal generator matrix is then*

$$G = [1 + \omega^{50} D + \omega^{12} D^2 + \omega^{147} D^3 + \omega^{114} D^4,$$
$$1 + \omega^{181} D + \omega^{79} D^2 + \omega^{29} D^3 + \omega^{67} D^4,$$
$$1 + \omega^{112} D + \omega^{235} D^2 + \omega^{226} D^3 + \omega^{191} D^4;$$
$$1 + \omega^{103} D + \omega^{155} D^2 + \omega^{45} D^3 + D^4,$$
$$1 + \omega^{67} D + D^2 + \omega^{160} D^3 + \omega^{88} D^4,$$
$$1 + \omega^{55} D + \omega^{177} D^2 + \omega^{183} D^3 + \omega^{161} D^4].$$

*The column distances are: $d_j^c = j + 2$ for $0 \leq j \leq 4$.*

# 4 Convolutional Codes With Finite Decoder Memory

In this section we will investigate the impact of finite decoder memory on the behaviour of convolutional codes in terms of symbol loss probability and symbol delay. The analysis is inspired by the method for calculating the exact symbol loss probability for deterministic block codes presented in [9].

## 4.1 Symbol Loss Probability

Let $\mathcal{C}$ be an $(n, k, \delta)$ convolutional code with generator matrix $G$ and memory $v$. The input to the encoding process is termed *symbols* and the output is termed *packets*. The transmission of a packet may succeed or result in an erasure. These two outcomes are modelled by 1 and 0, respectively. Let $r$ be the number of links in the logical network. Let $h$ be the number of packet transmissions, then the set $V_{h,r} \triangleq \left\{v_1, \ldots, v_{2^{hr}} \in \{0,1\}^{h\times r}\right\}$ contains all possible combinations of success and erasure for all the independent packet receptions. In a single hop network with one source and one sink then $r = 1$. Let $W \in V_{h,r}$ be the indicator matrix which contains a zero at element $(i, j)$ if the $(i, j)$'th packet is erased, and a 1 otherwise.

**Lemma C.3.** *Let $\bar{\epsilon} \in [0,1]^r$ be the erasure probability of the $r$ links in the network assuming all erasures are i.i.d., and let $W \in V_{h,r}$ be the indicator matrix. Then the probability of $W$ occurring is*

$$P(W|\bar{\epsilon}) = \prod_{i=1}^{h}\prod_{j=1}^{r}(1-\epsilon_j)\mathbb{1}\left[W_{i,j}=1\right] + \epsilon_j\mathbb{1}\left[W_{i,j}=0\right].$$

*Proof.* Due to the independence assumption it follows easily that the probability of $W$ occurring is the product of the probabilities of erasures and successful reception on the individual links. □

Let $\mathcal{C}$ be an $(n, k, \delta)$ convolutional code with memory $v$ and let $m_d = c\cdot(v+1)$ denote the memory of the decoder, where $c \in \mathbb{N}$. We now introduce $T_{L_C}(x, \mathcal{C}, W, c) \in \{0, 1\}$, which denotes the sink's ability to decode the $x$'th symbol given $\mathcal{C}$, $W$, and $m_d$, where $x \in \mathbb{N}_0$. That is, $T_{L_C}(x, \mathcal{C}, W, c)$ determine whether or not it is possible for the sink to decode the $x$'th symbol, when receiving the packets from $\mathcal{C}$, where successful reception is specified by the elements of $W$. It is trivial to evaluate $T_{L_C}(x, \mathcal{C}, W, c)$ by trying all combinations.

**Lemma C.4.** *Let $\mathcal{C}$ be an $(n, k, \delta)$ convolutional code and $\bar{\epsilon} \in [0,1]^r$ be the erasure probability of the $r$ links in the network, assuming all erasures are i.i.d. Assuming the sink has successfully decoded the symbols with indexes $\{-v-1, \ldots, -1\}$, where $v$ is the memory of the code, then the probability that the $x$'th symbol is not decodable is*

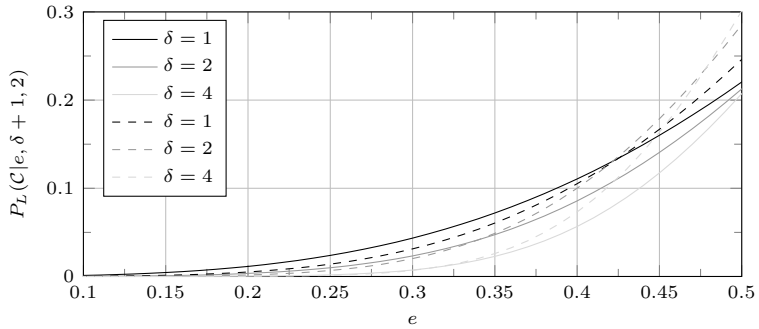



**Fig. C.2:** Evaluation of (C.3) using $(2, 1, \delta)$ convolutional codes. The dashed lines are ODP convolutional codes and the solid lines are random based codes.

$$P_L(\mathcal{C}|\overline{\epsilon}, x, c) = 1 - \sum_{W \in V_{\frac{n}{k}(x+m_d),r}} T_{L_C}(x, \mathcal{C}, W, c) P(W|\overline{\epsilon}). \tag{C.3}$$

*Proof.* For all elements of $V_{\frac{n}{k}(x+m_d),r}$ the probabilities of $W$ occurring where the $x$'th symbol is decodable are summed. The result of this summation is the probability that the $x$'th symbol is decodable. □

Fig. C.2 shows (C.3) evaluated at $x = \delta+1$ using $(2, 1, \delta)$ convolutional codes with memory $v = \delta$, and $c = 2$ in a single hop network. The figure includes three ODP convolutional codes where $\delta = 1, 2, 4$, and three random based convolutional codes with the same value of $\delta$. The ODP convolutional codes are constructed according to Lemma C.1 using the upper-left $(\delta + 1) \times (\delta + 1)$ submatrix of the two matrices in Ex. C.1. The random based codes also use the field $\mathbb{F}_{2^8}$, from which the coding coefficients are chosen at random [11]. The figure shows that ODP convolutional codes provide a symbol loss probability that is slightly lower than that of the random based codes. This is in fact the case over the entire range $0 \leq \epsilon \leq 1/2$.

The dependency on the decoding probability of previous symbols is implicitly captured by $T_{L_C}(x, \mathcal{C}, W, c)$. The average symbol loss probability across a countable infinite number of symbols can be calculated using (C.3) as

$$\lim_{N\to\infty} \frac{1}{N+1} \sum_{x=0}^{N} P_L(\mathcal{C}|\overline{\epsilon}, x, c). \tag{C.4}$$

## 4.2 Comparison of Infinite and Finite Decoder Memory

In the following we will compare the symbol loss probability of convolutional codes when the decoder has infinite or finite memory. Due to the considerable computational complexity of (C.4) we use Lemma C.4 instead.

Let $\mathcal{C}$ be a $(2,1,1)$ systematic convolutional code with $G = [1, 1+D]$ and let $r = 1$, then the authors of [12] derive a result equivalent to

$$\lim_{\substack{x\to\infty \\ c\to\infty}} P_L(\mathcal{C}|\epsilon, x, c) = \frac{\epsilon^3}{(1-\epsilon+\epsilon^2)^2}. \tag{C.5}$$

When evaluated at $\epsilon = 1/2$ and $\epsilon = 1/4$ then (C.5) is $\frac{2}{9} \approx 0.222222$ and $\frac{4}{169} \approx 0.023669$, respectively. It is of interest how this compares to the erasure probability when using finite decoder memory. Table C.1 lists values of (C.3) for different values of $c$ and $x$. From the table it is easily seen that the erasure probability converges towards that of (C.5), for increasing $c$ and $x$. In fact, for $\epsilon = 1/2$, $c = 5$, and $x = 8$ then $P_L(\mathcal{C}|\epsilon, x, c) = 0.222222$. By the same token, even for $\epsilon = 1/2$, $c = 2$, and $x = 8$ the gain of using infinite memory is less than $0.775\ \% = 1 - \frac{2/9}{0.223957}$.

For the sake of completeness we have also compared the convergence for another convolutional code, namely a $(2,1,2)$ systematic convolutional code with $G = [1, 1+D+D^2]$. For this convolutional code we observed a similar convergence for the symbol loss probability.

## 4.3 Symbol Delay

We now introduce $T_{D_C}(x, \mathcal{C}, W, c) \in \{0, \ldots, \frac{n}{k} m_d\}$, which denotes the number of packets transmitted from the encoder before the $x$'th symbol is decoded at the sink given $\mathcal{C}$ and $W$. We further define $T_{D_C}(x, \mathcal{C}, W, c) = 0$ if the $x$'th symbol is not decodable given $\mathcal{C}$ and $W$. We define symbol delay as the number of packets the source has transmitted, after receiving a symbol, before that symbol is decoded by the sink. It is trivial to evaluate $T_{D_C}(x, \mathcal{C}, W, c)$ by trying all combinations.

**Lemma C.5.** *Let $\mathcal{C}$ be an $(n,k,\delta)$ convolutional code and $\overline{\epsilon} \in [0,1]^r$. Assuming the sink has successfully decoded the symbols with indexes $\{-v-1, \ldots, -1\}$, where $v$ is the memory of the code, then the expected symbol delay (in terms of packets transmitted) for the $x$'th symbol is*

$$S_D(\mathcal{C}|\overline{\epsilon}, x, c) = \frac{\sum_{W \in V_{\frac{n}{k}(x+m_d),r}} T_{D_C}(x, \mathcal{C}, W, c) P(W|\overline{\epsilon})}{\sum_{W \in V_{\frac{n}{k}(x+m_d),r}} T_{L_C}(x, \mathcal{C}, W, c) P(W|\overline{\epsilon})}. \tag{C.6}$$

*Proof.* The numerator sums the amount of delay the $x$'th symbol endures, weighted by the probability of those delays occurring. The denominator sums the number of times the $x$'th symbol is decodable weighted by the probability of said symbol being decodable. Thus, the accumulated delay is divided by the total number of times the $x$'th symbol is decodable. □

Fig. C.3 shows the evaluation of (C.6) with the same parameters and codes as in Fig. C.2. The figure shows that ODP convolutional codes always provide

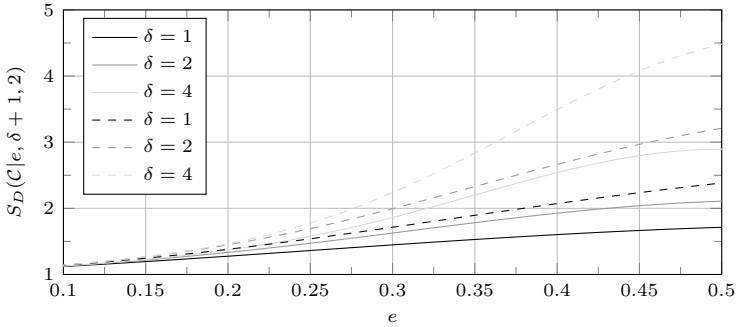



**Fig. C.3:** Evaluation of (C.6) using $(2, 1, \delta)$ convolutional codes. The dashed lines are ODP convolutional codes and the solid lines are random based codes.

**Table C.1:** Evaluation of (C.3) for $\epsilon = 1/2$, $\epsilon = 1/4$ and different $c$ and $x$.

| | $\epsilon = 1/2$ | | | $\epsilon = 1/4$ | | |
|---|---|---|---|---|---|---|
| $c$ | $x = 0$ | $x = 2$ | $x = 4$ | $x = 0$ | $x = 2$ | $x = 4$ |
| 1 | 0.18750 | 0.24609 | 0.24976 | 0.02734 | 0.03343 | 0.03365 |
| 2 | 0.16797 | 0.22046 | 0.22374 | 0.01952 | 0.02386 | 0.02401 |
| 3 | 0.16675 | 0.21886 | 0.22211 | 0.01924 | 0.02352 | 0.02368 |
| 4 | 0.16667 | 0.21876 | 0.22201 | 0.01923 | 0.02351 | 0.02366 |
| 5 | 0.16667 | 0.21875 | 0.22201 | 0.01923 | 0.02351 | 0.02366 |

the lowest symbol delay. Note that the difference in symbol delay between the ODP convolutional codes and the random based codes is mostly negligible.

There exists a trade-off between symbol loss probability and delay. Consider a rate $1/1$ systematic code. Such a code provides a constant delay of 1, regardless of the erasure rate of the channel, however, the symbol loss probability is equal to the erasure rate of the channel. Codes with a lower rate may produce a larger symbol delay but also a lower loss probability. Alternatively, consider a rate $1/2$ duplication code, this code provides a symbol delay of $\frac{2\epsilon+1}{\epsilon+1}$ and a symbol loss probability of $\epsilon^2$. That is, this code has a larger symbol delay since more symbols may be received later than using the previous code, and thus it also has a lower symbol loss probability.

# 5 Conclusions

In this paper we presented explicit constructions for any $(n, k, \delta)$ ODP convolutional code, based on jointly superregular lower triangular Toeplitz matrices. For convolutional codes with finite decoder memory we provided a method to calculate the exact symbol loss probability and delay for each symbol individually. This method is general in the sense that it is applicable to any $(n, k, \delta)$

convolutional code. For two specific systematic convolutional codes we compared the symbol loss probability for the two scenarios where the sink has either finite or infinite memory. These comparisons showed that for $c \geq 2$ the symbol loss probability is not significantly affected by the use of a finite amount of decoder memory.

We showed that ODP convolutional codes produce a lower symbol loss probability than random based codes, regardless of the erasure probability of the channel. Furthermore, we showed that ODP convolutional codes consistently provide a lower symbol loss probability and delay over random based convolutional codes. Both of these two benefits are rooted in the fact that these static codes have predictable performance, given that they are not based on random coefficients.

# References

[1] R. Roth and A. Lempel, "On MDS codes via Cauchy matrices," *IEEE Transactions on Information Theory*, vol. 35, no. 6, pp. 1314–1319, November 1989.

[2] H. Gluesing-Luerssen, J. Rosenthal, and R. Smarandache, "Strongly-MDS convolutional codes," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 584–598, February 2006.

[3] P. Almeida, D. Napp, and R. Pinto, "A new class of superregular matrices and MDP convolutional codes," *Linear Algebra and its Applications*, vol. 439, no. 7, pp. 2145–2157, 2013.

[4] R. Hutchinson, R. Smarandache, and J. Trumpf, "On Superregular Matrices and MDP Convolutional Codes," *Linear Algebra and its Applications*, vol. 428, no. 11-12, pp. 2585–2596, 2008.

[5] P. Almeida, D. Napp, and R. Pinto, "Superregular matrices and applications to convolutional codes," *Linear Algebra and its Applications*, vol. 499, pp. 1–25, June 2016.

[6] P. Elias, "Coding for noisy channels," *IRE Convention Records Part 4*, pp. 37–46, 1955.

[7] M. Arai, A. Yamamoto, A. Yamaguchi, S. Fukumoto, and K. Iwasaki, "Analysis of using convolutional codes to recover packet losses over burst erasure channels," *IEEE Pacific Rim International Symposium on Dependable Computing*, pp. 258–265, 2001.

[8] R. Johannesson and K. Zigangirov, *Fundamentals of Convolutional Coding*, 2nd ed. Wiley, 2015.

[9] J. Hansen, J. Østergaard, J. Kudahl, and J. H. Madsen, "Superregular lower triangular toeplitz matrices for low delay wireless streaming," *IEEE Transactions on Communications*, vol. 65, no. 9, pp. 4027–4038, September 2017.

[10] R. Mahmood, A. Badr, and A. Khisti, "Convolutional codes with maximum column sum rank for network streaming," *IEEE International Symposium on Information Theory*, pp. 2271–2275, June 2015.

[11] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," *IEEE International Symposium on Information Theory*, June 2003.

[12] B. M. Kurkoski, P. H. Siegel, and J. K. Wolf, "Exact probability of erasure and a decoding algorithm for convolutional codes on the binary erasure channel," *IEEE Global Telecommunications Conference*, vol. 3, pp. 1741–1745, December 2003.

# Paper D

## When are Erasure Correcting Block Codes Better than Convolutional Codes in a Multi-hop Network?

Jonas Hansen, Jan Østergaard, Johnny Kudahl, and John H. Madsen

The paper has been published in the proceedings of the
*11th International Conference on Signal Processing and Communication Systems*, 2017.

*The layout has been revised.*

# Abstract

*In this paper we investigate the effect of imposing a maximum allowed delay on the symbol loss probability for a set of rate $1/2$ erasure correcting codes. Given some maximum allowable delay, we define the effective symbol loss probability to be the probability that a symbol is received too late or not at all. Consider a network with three nodes; source, relay, and sink. The source encodes data using an erasure correcting code, the relay decodes, recodes, and finally the sink decodes using Gaussian elimination. We compare the effective symbol loss probability of systematic triangular block codes, dense block codes, and systematic convolutional codes. For a wide range of packet loss probabilities and allowable symbol delays, our results show that the systematic triangular block codes are superior. Our results also show that the field size does not affect the gain in effective symbol loss probability.*


# 1 Introduction

The amount of real-time streaming of audio and video is ever increasing [1]. Low delay streaming of high resolution audio and video to several receivers is problematic even for modern wireless networks [2]. A way to overcome this may be to use some type of forward erasure correction (FEC). One immediate advantage of FEC codes, in general, is that they enable a trade-off between bit rate, delay and loss probability [3]. Furthermore, for streaming applications with strict play-out timing constraints such as music streaming, an FEC code can even help in reducing the play-out buffer size at the decoder [4].

Erasure correcting codes can be used to simplify coordination when used with routing [5], and this can be utilized to increase the throughput of various transport protocols [6]. In [7] the authors proposed a network coding scheme that significantly increased throughput and reduced latency for reliable transport protocols such as TCP. However, designing FEC codes for arbitrary cases (e.g., networks, loads, and protocols) is non-trivial. The authors of [8] proposed that the coding coefficients are chosen at random over some finite field. Using random coding coefficients has also proved efficient for networks with recoding. Networks where coding is not only performed at the end nodes but also at intermediate nodes in the network is termed network coding [9]. Network coding can be performed on a set of source symbols. If this set has a finite and fixed size it is termed a block code [10]. One of the benefits of a finite size is that they can be studied using an exhaustive search, e.g., where all possible erasure patterns are compared. The coding matrices used to construct block codes can have different structures. In [11] the authors presented codes based on coding matrices with a lower triangular Toeplitz structure. This reduces the amount of non-zero coding coefficients, and was shown to increase coding

throughput substantially. Furthermore, codes based on triangular matrices inherit a low delay property, since encoding can be performed immediately as new symbols are received by the source. The triangular structure trades-off symbol loss probability for a lower symbol delay and allows for unequal erasure protection of the symbols. That is, the first symbol in a triangular block code has the smallest symbol loss probability, the second symbol has slightly larger (depending on the block size) symbol loss probability, and so forth.

In [12] the authors proposed a family of codes based on dense coding matrices. This family of dense codes have the benefit of being maximum distance separable (MDS), which the triangular codes do not. However, dense codes do not possess the low delay property since the entire input vector needs to be available when encoding. Additionally, these dense codes (over sufficiently large fields) also exercise the all-or-nothing principle, i.e., either all symbols are fully decoded or no symbols are decoded at all.

If coding is not confined to a block of symbols but used in a continuous manner, then it is termed a convolutional code [13]. At the encoder, convolutional codes possess the same low delay property that triangular block codes do. Convolutional codes are, however, not as easy to study as block codes, since they may consist of a countable infinite number of symbols. For convolutional codes we show that the continuous nature of the code words combined with recoding may introduce code words with data dependencies that are considerably longer than the code words the encoder emits. Moreover, this longer dependency may then increase the delay.

A lot of research efforts have been put into analysing the loss and delay performance for different kind of codes. In [14] the authors compared the performance of short block length low-density parity-check (LDPC) codes and convolutional codes. The average delay performance of block coding schemes when the arrival stream is stochastic is analysed in [15]. Finally, the authors of [16] developed a queueing model for discrete memoryless channels with block coding and feedback. In this paper we consider forward erasure correcting codes. That is, codes that do not require feedback, therefore we have not investigated methods such as instantly decodable network coding [17].

In this paper we evaluate and compare the probability of receiving symbols within some time limit for three families of codes. First we provide some background on how we perform the coding and define the performance metrics used. In Section 4 we cover the simulations method which is the basis for the result discussed in Section 5.

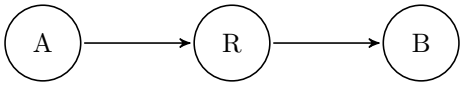



**Fig. D.1:** Alice is sending packets to Bob, via the Relay.

# 2 Background

We now demonstrate how encoding is carried out for block codes. Let $n, k, l \in \mathbb{N}$ and let $A$ be an $n \times k$ coding matrix, this yields a code with rate $k/n$. The messages to be encoded using $A$ are stored in $S$, a $k \times l$ matrix, known as the source data matrix. The result of the encoding process is $C$, an $n \times l$ matrix, the coded data matrix. Therefore $C = AS$, which shows how the $k$ rows (of length $l$) of the source data matrix are encoded into the $n$ rows (of length $l$) of the coded data matrix. We define each row of the source data matrix as a *symbol* and each row of the coded data matrix as a *packet*. The vector used to encode a packet is defined as the packet's coding vector. The number of non-zero coefficients in a coding vector is defined as the code length. A packet is said to be systematic if it has a code length of one. For convolutional codes, two coding vectors are said to be overlapping if the intersection of their support[1] is not the empty set. That is, let $c_1$ and $c_2$ be two coding vectors, then they are overlapping if and only if

$$\text{supp}(c_1) \cap \text{supp}(c_2) \neq \emptyset,$$

where $\text{supp}(c)$ is the support of a coding vector $c$.

We consider the network shown in Fig. D.1. The network consists of three nodes; source (A), relay (R), and sink (B). The erasures on the two links are assumed to be i.i.d. The source receives (or generates) symbols at some fixed rate and it outputs two packets for each symbol, thus it is a rate $1/2$ code. One benefit of this network is that it is sufficiently simple to be used as a building block for larger and more complex networks.

The relay and the sink each maintain a matrix with the received packets (and their coding vectors). When a packet is received it is augmented (including the coding vector) to this decoding matrix, which thereby increases the dimensions of the matrix. Gaussian elimination is then performed, in order to evaluate if the newly added packet is linearly independent of the other packets in the decoding matrix. That is, the node attempt to decode the packets by using Gaussian elimination on the coding vectors. This paper considers network codes where the relaying node performs decoding.

# 3 Symbol Loss Probability and Delay

The set of erasure correcting codes we have investigated is; dense block codes, triangular block codes, and convolutional codes. The triangular block codes and the convolutional codes are systematic, whereas the dense codes are not. The encoding, recoding, and decoding procedure is as follows. The source

[1] The support of a vector is the indices of non-zero elements in the vector.

encodes packets (when possible) and forwards them to the relay. When the relay receives a packet it will decode it, and if the packet is linearly independent of the other packets in the decoding matrix, the relay recodes and transmits two packets. The two recoded packets are forwarded to the sink, which then decodes all the received packets using Gaussian elimination. When using a dense code, the source does not transmit any packets before all the symbols of a block are received.

**Definition D.1 (Symbol Delay).** *When the sink decodes symbol $j$ and the source contains $i$ symbols, then the delay experienced by symbol $j$ is $i - j$. The delay experienced by undecodable symbols is not defined.* △

**Definition D.2 (Effective Symbol Loss).** *Let $d$ be the maximum allowable symbol delay. That is, any symbol that experiences a delay larger than $d$ is considered lost. Finally, the probability of a symbol being lost, either due to a large delay or erasures, is termed the effective symbol loss.* △

In Section 5 the symbol loss probability and symbol delay, will be used to determine which code type (and which parameters) yields the lowest effective symbol loss probability given some maximum allowed delay. From Definition D.1 it is clear that systematic packets experience a delay of 0 (if received by the sink), since they are transmitted immediately when received by the source. Thus, the channel latency is not taken into account. Furthermore, it is assumed that the channel is not saturated.

Let $\mathcal{C}$ be an erasure correcting code with rate $1/2$ and constraint length $v \in \mathbb{N}$. The constraint length is the maximum code length, i.e., the maximum number of non-zero coefficients in a coding vector. In this work we consider two code types, namely block codes and convolutional codes. Thus, the code $\mathcal{C}$ is either a $(2v, v)$ block code or a $(2, 1, v-1)$ convolutional code. For convolutional codes we define the degree (third parameter) of $\mathcal{C}$ as the sum of the row degrees of the basic minimal generator matrix [18]. Furthermore, we define $m \in \mathbb{N}$ as the memory factor that together with $v$ determines the amount of packets the relay and sink may have in memory. That is, the relay and sink may store, at most, $mv$ packets at any time. Note that only $m = 1$ is applicable to block codes, and we shall therefore only consider different values of $m$ for convolutional codes. Moreover, the coding vectors of the $mv$ packets are required to uphold the following

$$\min(\text{supp}(c_i)) - \min(\text{supp}(c_j)) < mv, 1 \leq j < i \leq mv, \tag{D.1}$$

where $c_i$ and $c_j$ are the $i$th and $j$th rows (coding vectors) of the decoding matrix. If this is not the case then the $j$th packet is removed from the decoding matrix. The condition on $i$ and $j$ come from the decoding process, which ensures that any two coding vectors do not have identical sup-

port: $\text{supp}(c_i) \neq \text{supp}(c_j), i \neq j$ and that the coding vectors are ordered:

$$\min(\text{supp}(c_i)) > \min(\text{supp}(c_j)), 1 \leq j < i \leq mv.$$

For dense codes, the code length is the same for all coded symbols, from both the encoder and the relay, namely $v$. For the triangular codes, the code length is also the same from both the encoder and the relay. However, since the code is triangular $v$ defines the maximum code length. That is, for triangular codes, $v$ is the code length of the last packet in a block. Finally, for convolutional codes $v$ is the code length for all non-systematic packets, except the first $v-1$. The relay on the other hand, is not limited to a code length of $v$. This is due to the continuous nature of the convolutional code.

**Lemma D.1.** *Let $\mathcal{C}$ be a $(2, 1, v-1)$ convolutional code, where $v$ is the maximum code length from the encoder and let $m$ be the memory factor of the relay. Then the maximum code length from the relay, $v_r$, is:*

$$\begin{aligned} v_r &= \min(v^2, (m+1)v - 1)) \\ &= \begin{cases} v^2, & m \geq v, \\ (m+1)v - 1, & m < v. \end{cases} \end{aligned}$$

*Proof.* For $m \geq v$ it is obvious that combining $v$ non-overlapping coding vectors, each with a maximum code length of $v$, results in a maximum code length of $v^2$. For $m < v$ it follows from (D.1) that the first $v-1$ coding vectors can have a code length of at most $mv - 2 + v$, and that the last ($v$th) coding vector can contribute at most 1. Thus, the sum is: $mv + v - 1 = (m+1)v - 1$. □

**Definition D.3 (Symbol Delay Distribution).** *A symbol delay distribution is a probability mass function, which states that the probability that a decodable symbol will experience a delay of $d$ is $f_D(d; e, \mathcal{C}, m) = \Pr(D = d \mid e, \mathcal{C}, m)$ given some erasure probability $e$, code $\mathcal{C}$, and memory factor $m$, where $D \in \mathbb{N}_0$.* △

Let $e \in [0, 1]$ be the erasure probability, $\mathcal{C} \in \text{GF}(q)^{n \times k}$, and $m \in \mathbb{N}$. Then the probability that a symbol experiences a delay less than or equal to $d$ is given by the cumulative distribution function of the delay, i.e.

$$\Pr(D \leq d \mid e, \mathcal{C}, m) = \sum_{i=0}^{d} f_D(i; e, \mathcal{C}, m).$$

**Lemma D.2.** *Let $e \in [0, 1]$ be the erasure probability, and $\mathcal{C} \in \text{GF}(q)^{n \times k}$, and $m \in \mathbb{N}$. Then, given a symbol loss probability $P_L(e, \mathcal{C}, m)$ and a maximum allowable delay $d$, then the effective symbol loss probability is given by:*

$$P_E(d; e, \mathcal{C}, m) = 1 - \Pr(D \leq d \mid e, \mathcal{C}, m) \cdot (1 - P_L(e, \mathcal{C}, m)). \tag{D.2}$$

*Proof.* $1 - P_L(e, \mathcal{C}, m)$ is the probability of successful reception of a symbol. This probability is multiplied with the probability that a symbol experiences an allowable delay, which yield the probability that a symbol is both received and experiences an allowable delay. □

# 4 Simulation Method

A delay distribution was obtained through simulations of each set of values of $v$ and $e$ (and $m$ for convolutional codes). In order to ensure that the simulations use the exact same amount of symbols for all three code types, we used $\left\lceil \frac{10^6}{v} \right\rceil v$ symbols in each simulation. The simulations used the erasure probabilities $e \in \{0.05, 0.10, \ldots, 0.45\}$, and the erasure pattern was identical for all three code types. Each of the code types were simulated with different constraint lengths $v \in \{1, \ldots, 64\}$. For the convolutional codes the relay and sink are restricted to having, at most, $mv$ symbols in memory, where $m \in \{1, \ldots, 4\}$. The coding coefficients were chosen at random from the finite field $\mathrm{GF}(2^8)$, without using the element 0.

The result of these simulations is an empirical delay distribution for each set of simulation parameters, code type, and set of code parameters. Then for each pair of $e$ and $d \in \{0, \ldots, 15\}$, the effective symbol loss probability is calculated for all delay distributions using (D.2). Finally, the code type and corresponding parameters yielding the lowest effective symbol loss probability is found for each pair of $e$ and $d$.

Let the *best code* be the code that yields the lowest effective symbol loss probability. Let $p_1$ be the effective symbol loss probability of the best code, and $p_2$ be the effective symbol loss probability for the second best code. Then the gain in effective symbol loss probability is defined as:

$$g_l \triangleq \frac{1 - p_1}{1 - p_2} - 1.$$

# 5 Discussion

Fig. D.2 shows part of a delay distribution for the three code types under some specified code parameters and channel conditions. From the figure it becomes evident that the delay distribution for dense codes can be approximated with a uniform distribution. In fact, our experiments show that this is generally the case when $e < .35$. On the other hand, the delay distributions for triangular and convolutional codes share a similar structure. Their structure is primarily a peak in $d = 0$ and a low tail. For the case $v = 1$ all three code types performs identically.

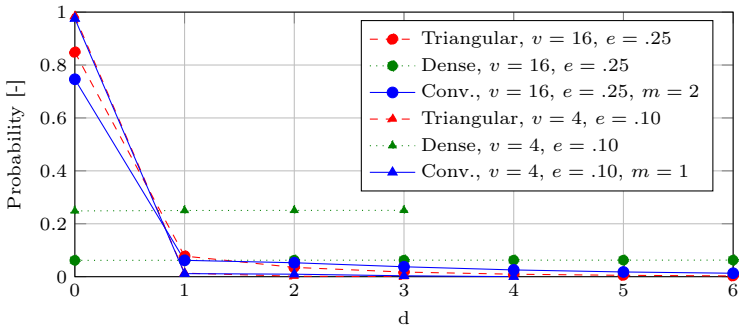



**Fig. D.2:** Examples of delay distributions. The full support of the distributions is not shown for $v = 16$.

The gain in effective symbol loss of the best code type over the second best code type is shown in Table D.2a on page 127. The code parameters yielding the lowest effective symbol loss is shown in Table D.2b. In all scenarios where either a convolutional code or dense code was the best performing code, a triangular code performed almost equally as well. That is, the gain in terms of effective symbol loss probability of using a convolutional code or dense code is at most 0.28 % and 0.96 %, respectively. However, a triangular code outperformed the convolutional or dense codes with 5.38 % ($d = 11$, $e = 0.30$) and 11.19 % ($d = 15$, $e = 0.40$), respectively.

## 5.1 Small Matrices and Low Delay

In [11] we proposed the use of lower triangular coding matrices with static coding coefficients over small dimensions, e.g., $10 \times 10$, for low delay streaming. The benefits of these matrices are multifold: 1) lower complexity for both encoding and decoding, e.g., decoding using Gaussian elimination has cubic algorithmic complexity, 2) increased coding throughput performance, by carefully choosing the coding coefficients, 3) predictable performance given that the coding coefficients are constant. Table D.1a on the following page shows the gain in effective symbol loss when $v \in \{1, \ldots, 10\}$, and the code parameters are shown in Table D.1b. From the table it is clear that the structure is preserved, compared to Table D.2a, despite the shorter constraint length.

## 5.2 Field Size Bias

For completeness a set of simulations using the finite field GF($2^{32}$) was also conducted. The coding coefficients were again chosen at random from the finite field without using the element 0. This set of simulations is used to eliminate any bias in the results, depending on the field size. The simulations were run

**Table D.1:** Gain, in percent, of the best code type over the second best code type, using GF($2^8$) and $v \in \{1, \ldots, 10\}$. The colour/font of the cell indicates the code type. Red/bold: triangular code, green/italic: dense code, blue/normal: convolutional code, and white: $v = 1$.

**(a)**

| e \ d | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|---|
| **0.05** | 0 | **0.09** | 0.01 | 0.05 | 0.06 | 0.05 | 0.05 | 0.04 |
| **0.10** | 0 | **0.33** | **0.13** | 0.15 | 0.24 | 0.27 | 0.26 | 0.23 |
| **0.15** | 0 | **0.56** | **0.56** | 0.06 | 0.43 | 0.63 | 0.68 | 0.65 |
| **0.20** | 0 | **0.45** | **0.93** | **0.77** | 0.08 | 0.68 | 1.12 | 1.27 |
| **0.25** | 0 | *0.28* | **1.07** | **1.25** | **0.94** | **0.66** | 0.29 | 0.97 |
| **0.30** | 0 | *0.89* | **0.27** | **1.14** | **1.21** | **1.05** | **0.67** | **0.20** |
| **0.35** | 0 | *0.89* | *0.96* | **0.29** | **1.06** | **1.29** | **1.11** | **0.75** |
| **0.40** | 0 | *0.46* | *0.77* | *0.70* | *0.05* | **0.83** | **1.03** | **1.16** |
| **0.45** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**(b)**

| e \ d | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|---|
| **0.05** | 1 | **10** | 3,1 | 5,4 | 10,4 | 10,1 | 10,1 | 8,1 |
| **0.10** | 1 | **10** | **10** | 4,1 | 8,3 | 8,3 | 7,1 | 10,1 |
| **0.15** | 1 | **10** | **10** | 4,1 | 5,1 | 10,2 | 10,2 | 10,2 |
| **0.20** | 1 | **10** | **10** | **10** | 4,4 | 5,4 | 5,4 | 6,2 |
| **0.25** | 1 | *2* | **10** | **10** | **10** | **10** | 4,2 | 4,2 |
| **0.30** | 1 | *2* | **9** | **10** | **10** | **10** | **10** | **10** |
| **0.35** | 1 | *2* | *3* | **10** | **10** | **10** | **10** | **10** |
| **0.40** | 1 | *2* | *3* | *4* | *5* | **10** | **10** | **10** |
| **0.45** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

with the same number of symbols and erasure pattern as the other simulations. Table D.2c on the next page shows the gain in effective symbol loss for this set of simulations. From the table it is clear that the result does not significantly depend on the field size, given that the values in Table D.2a and D.2c are nearly identical.

# 6 Conclusions

Triangular block codes outperform both dense block codes and convolutional codes in most scenarios. Furthermore, in the few cases that triangular block codes do not have the best performance they are second best with a negligible margin. This result does not depend on the size of the used field, since there is little to no difference between using the fields GF($2^8$) and GF($2^{32}$)

For applications that require a low complexity, e.g., on embedded platforms, limiting the code structure to only matrices of small dimensions does not significantly change the ranking between the code types, for different values of $e$ and $d$. The method described in this paper to obtain delay distributions can easily be extended to other code types, code structures, and parameters.

**Table D.2:** (a) and (c) show the gain, in percent, of the best code type over the second best code type, using GF($2^8$) and GF($2^{32}$), respecively. (b) shows the parameters of (a). The parameter is the constraint length $v$, and for the convolutional code $m$ is the second parameter Red/bold: triangular, green/italic: dense, blue/normal: convolutional, and white: $v = 1$.

**(a)**

| e \ d | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0.05** | 0 | **0.13** | **0.04** | **0.00** | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 |
| **0.10** | 0 | **0.48** | **0.34** | **0.09** | **0.00** | 0.03 | 0.04 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 |
| **0.15** | 0 | **0.83** | **1.07** | **0.57** | **0.25** | **0.08** | 0.02 | 0.07 | 0.10 | 0.11 | 0.12 | 0.13 | 0.13 | 0.13 | 0.13 | 0.13 |
| **0.20** | 0 | **0.74** | **1.78** | **1.97** | **1.32** | **0.85** | **0.50** | **0.25** | **0.08** | 0.03 | 0.11 | 0.17 | 0.21 | 0.24 | 0.26 | 0.28 |
| **0.25** | 0 | *0.28* | **2.00** | **2.95** | **3.20** | **3.34** | **2.68** | **2.19** | **1.75** | **1.34** | **1.01** | **0.72** | **0.50** | **0.31** | **0.14** | **0.01** |
| **0.30** | 0 | *0.89* | **0.41** | **2.66** | **3.92** | **4.76** | **5.20** | **5.39** | **5.59** | **5.65** | **5.53** | **5.38** | **5.06** | **4.80** | **4.35** | **3.95** |
| **0.35** | 0 | *0.89* | *0.96* | **0.29** | **2.03** | **4.07** | **5.49** | **6.56** | **7.27** | **7.91** | **8.26** | **8.48** | **8.74** | **8.75** | **8.81** | **8.93** |
| **0.40** | 0 | *0.46* | *0.77* | *0.70* | *0.05* | **0.92** | **1.68** | **2.97** | **4.29** | **5.62** | **6.94** | **8.10** | **8.98** | **9.93** | **10.46** | **11.19** |
| **0.45** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **0.46** | **1.11** |

**(b)**

| e \ d | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| .05 | 1 | **60** | **60** | **60** | 12,2 | 48,1 | 27,2 | 8,1 | 8,1 | 8,1 | 8,1 | 8,1 | 8,1 | 8,1 | 8,1 | 8,1 |
| .10 | 1 | **60** | **60** | **62** | **62** | 21,4 | 14,3 | 46,4 | 50,2 | 50,2 | 11,1 | 11,1 | 11,1 | 11,1 | 11,1 | 11,1 |
| .15 | 1 | **62** | **62** | **62** | **62** | **62** | 23,1 | 23,1 | 60,1 | 41,4 | 54,1 | 54,1 | 44,4 | 44,4 | 44,4 | 44,4 |
| .20 | 1 | **63** | **63** | **63** | **63** | **63** | **63** | **63** | **63** | 19,4 | 19,4 | 19,4 | 22,4 | 37,4 | 19,1 | 19,1 |
| .25 | 1 | *2* | **60** | **60** | **63** | **64** | **64** | **64** | **64** | **64** | **64** | **64** | **64** | **64** | **64** | **64** |
| .30 | 1 | *2* | **16** | **64** | **64** | **64** | **64** | **64** | **64** | **64** | **64** | **64** | **64** | **64** | **64** | **64** |
| .35 | 1 | *2* | *3* | **10** | **61** | **61** | **61** | **62** | **62** | **62** | **62** | **62** | **64** | **64** | **64** | **64** |
| .40 | 1 | *2* | *3* | *4* | *5* | **11** | **16** | **23** | **31** | **60** | **60** | **60** | **64** | **64** | **64** | **64** |
| .45 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | **27** | **27** |

**(c)**

| e \ d | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0.05** | 0 | **0.13** | **0.04** | 0.00 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 |
| **0.10** | 0 | **0.48** | **0.34** | **0.09** | **0.00** | 0.03 | 0.04 | 0.04 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 |
| **0.15** | 0 | **0.83** | **1.07** | **0.56** | **0.25** | **0.08** | 0.02 | 0.07 | 0.10 | 0.11 | 0.12 | 0.13 | 0.13 | 0.13 | 0.13 | 0.13 |
| **0.20** | 0 | **0.73** | **1.76** | **1.95** | **1.31** | **0.83** | **0.48** | **0.24** | **0.07** | 0.04 | 0.12 | 0.18 | 0.22 | 0.24 | 0.27 | 0.28 |
| **0.25** | 0 | *0.28* | **1.98** | **2.92** | **3.16** | **3.29** | **2.65** | **2.16** | **1.73** | **1.32** | **0.99** | **0.70** | **0.47** | **0.28** | **0.12** | 0.01 |
| **0.30** | 0 | *0.91* | **0.38** | **2.63** | **3.87** | **4.70** | **5.12** | **5.31** | **5.51** | **5.56** | **5.45** | **5.32** | **5.02** | **4.77** | **4.31** | **3.92** |
| **0.35** | 0 | *0.93* | *0.99* | **0.28** | **1.99** | **4.01** | **5.43** | **6.49** | **7.21** | **7.83** | **8.18** | **8.39** | **8.66** | **8.64** | **8.68** | **8.80** |
| **0.40** | 0 | *0.50* | *0.82* | *0.76* | *0.10* | **0.90** | **1.63** | **2.94** | **4.24** | **5.58** | **6.88** | **8.00** | **8.91** | **9.84** | **10.37** | **11.10** |
| **0.45** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **0.50** | **1.16** |

# References

[1] S. A. Hosseini, Z. Lu, G. de Veciana, and S. S. Panwar, "SVC-Based Multi-User Streamloading for Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 8, pp. 2185–2197, August 2016.

[2] S. Hahm, P. Kang, H. Bang, and H. Yeon, "Dynamic media buffer control scheme for seamless streaming in wireless local area networks," *IEEE Wireless Communications and Networking Conference Workshops*, pp. 109–114, 2016.

[3] M. Muntner and J. Wolf, "Predicted performance of error-control techniques over real channels," *IEEE Transactions on Information Theory*, vol. 14, no. 5, pp. 640–650, September 1968.

[4] J. H. Sørensen, P. Popovski, and J. Østergaard, "Delay minimization in real-time communications with joint buffering and coding," *IEEE Communications Letters*, vol. 21, no. 1, pp. 52–55, January 2017.

[5] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 169–180, August 2007.

[6] J. Krigslund, J. Hansen, D. E. Lucani, F. H. P. Fitzek, and M. Médard, "Network coded software defined networking: Design and implementation," *21th European Wireless Conference*, pp. 1–6, May 2015.

[7] J. Hansen, J. Krigslund, D. E. Lucani, and F. H. P. Fitzek, "Sub-Transport Layer Coding: A Simple Network Coding Shim for IP Traffic," *IEEE 80th Vehicular Technology Conference*, pp. 1–5, 2014.

[8] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," *IEEE International Symposium on Information Theory*, June 2003.

[9] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[10] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, April 1950.

[11] J. Hansen, J. Østergaard, J. Kudahl, and J. H. Madsen, "Superregular lower triangular toeplitz matrices for low delay wireless streaming," *IEEE Transactions on Communications*, vol. 65, no. 9, pp. 4027–4038, September 2017.

[12] I. S. Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[13] P. Elias, "Coding for noisy channels," *IRE Convention Records Part 4*, pp. 37–46, 1955.

[14] C. Rachinger, J. B. Huber, and R. R. Müller, "Comparison of Convolutional and Block Codes for Low Structural Delay," *IEEE Transactions on Communications*, vol. 63, no. 12, pp. 4629–4638, December 2015.

[15] R. N. Swamy and T. Javidi, "Delay analysis of block coding over a noisy channel with limited feedback," *42nd Asilomar Conference on Signals, Systems and Computers*, pp. 1431–1435, October 2008.

[16] R. Lübben and M. Fidler, "On the delay performance of block codes for discrete memoryless channels with feedback," *IEEE 35th Sarnoff Symposium*, pp. 1–6, May 2012.

[17] S. Sorour and S. Valaee, "On minimizing broadcast completion delay for instantly decodable network coding," *IEEE International Conference on Communications*, pp. 1–5, May 2010.

[18] H. Gluesing-Luerssen, J. Rosenthal, and R. Smarandache, "Strongly-MDS convolutional codes," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 584–598, February 2006.

# Paper E

## Sequential use of Block Codes and Convolutional Codes in a Real-Time Multi-hop Network

Jonas Hansen, Jan Østergaard, Johnny Kudahl, and John H. Madsen

The paper has been submitted to the
*IEEE 87th Vehicular Technology Conference*, 2018.

# Abstract

*Real-time streaming of audio and video have tight constraints on the delay allowed at each step in the distribution chain. In this paper, we focus on the transport layer. That is, we investigate the effect of imposing a maximum allowable delay on the symbol loss probability for a set of rate $1/2$ erasure correcting codes. We define the effective symbol loss probability to be the probability that a symbol is received too late or not at all, given some maximum allowable delay. We consider networks where the source and sink communicate via a relaying node, and that this intermediate node performs recoding. The erasure correcting code used between the source and relay need not be the same as the erasure correcting code used between the relay and sink. Moreover, on the first link, we use both a block code and on the second link we use a convolutional code. In order to do a fair comparison, we also include the case where both links use the same code type. Our results show that in order to minimize the effective symbol loss, the first link should use a triangular block code. Whereas, the second link should use a convolutional or triangular code, with little to no gain of using the former over the latter.*


# 1 Introduction

Streaming of audio and video has become an integrated part of the lives of many. Streaming high resolution audio or video over a wireless network to multiple receivers in non-trivial, especially if there are strict delay constraints, e.g, for live streaming [1]. To this end, a number of transport strategies [2–4], coding schemes [5–7], and source coding methods [8–10] have been researched.

Forward erasure correcting codes may be used sequentially in separate groups [11] or in a continuous manner. The former is termed block codes, and the latter is termed convolutional codes [12]. For block codes, we focus on two distinct forms, namely dense codes and triangular codes. Dense codes are so termed since their encoding matrix is fully dense. Triangular codes are constructed using lower triangular encoding matrices. Dense codes try to minimize the overall symbol loss. Whereas, triangular codes utilize a trade-off between loss and delay. On the other hand, convolutional codes seek to ensure a low encoding delay while also minimizing the symbol loss probability, at the cost of a larger complexity.

It is termed network coding [13] when coding is not only performed at the end nodes but also at intermediate nodes in the network. In this paper, we focus on the use of network codes when minimizing the symbol loss given strict delay constraints. The authors of [14] proposed that the coding coefficients are chosen at random over some finite field, and in this paper, we focus on the field $\mathrm{GF}(2^8)$. Using random coding coefficients has been shown to be efficient

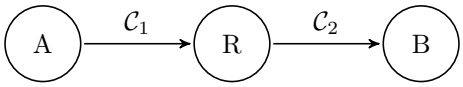



**Fig. E.1:** Alice is sending packets to Bob, via the Relay.

for networks with recoding [15]. We do not restrict the source and relay to use the same code type nor constraint length. The work presented in this paper extends the result presented in [16] by the authors of the present paper, where the source and relay would use the same code and constraint length.

## 2 Background

We consider the network shown in Fig. E.1. The network consists of three nodes; source (A), relay (R), and sink (B). The erasures on the two links are assumed to be independent and Bernoulli distributed. Rate $1/2$ erasure correcting codes are employed on the two links, namely $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively. The codes $\mathcal{C}_1$ and $\mathcal{C}_2$ have the constraint lengths $v_1$ and $v_2$, respectively. The code $\mathcal{C}_j, j \in \{1, 2\}$ is either a $(2v_j, v_j)$ block codes or a $(2, 1, v_j - 1)$ convolutional codes. For convolutional codes, we define the degree (third parameter) of $\mathcal{C}_j$ as the sum of the row degrees of the basic minimal generator matrix [17]. The code types that have been investigated are listed in Table E.1. For the code types TT and DD the constraint lengths of $\mathcal{C}_1$ and $\mathcal{C}_2$ are the same, that is $v_1 = v_2$.

For block codes, the encoding process is as follows. Let $n, k, h \in \mathbb{N}$ and let $A$ be an $n \times k$ encoding matrix, naturally, this constructs a code with rate $k/n$. The source data matrix $S$ is a $k \times h$ matrix which contains the data that is to be encoded. The coded data matrix $C$ is an $n \times h$ matrix that is the result of the encoding process. That is, $C = AS$, which shows how the $k$ rows (of length $h$) of the source data matrix are encoded into the $n$ rows (of length $h$) of the coded data matrix. We define each row of the source data matrix as a *symbol* and each row of the coded data matrix as a *packet*. Each row of the encoding matrix is denoted the coding vector of the corresponding packet of the coded data matrix. The code length of a coding vector is the size of the support[1] of the coding vector. If a coding vector has a code length of one, then the corresponding packet is defined as systematic. Two coding vectors are said to be overlapping if the intersection of their support is not the empty set. That is, let $c_1$ and $c_2$ be two coding vectors, then they are overlapping if and only if

$$\text{supp}(c_1) \cap \text{supp}(c_2) \neq \emptyset,$$

where $\text{supp}(c)$ is the support of a coding vector $c$.

[1]The support of a vector is the indices of the non-zero elements of the vector.

**Table E.1:** Alias' for the examined code types.

| Alias | Code type of $\mathcal{C}_1$ | Code type of $\mathcal{C}_2$ |
|---|---|---|
| CC | Systematic convolutional | Convolutional |
| TC | Systematic triangular block | Convolutional |
| TT | Systematic triangular block | Triangular block |
| DD | Dense block | Dense block |
| DC | Dense block | Convolutional |

The relay and sink each holds a matrix, where all the received packets and corresponding coding vectors are stored. This matrix is denoted the decoding matrix. Every time a packet is received it is, together with the coding vector, augmented to the decoding matrix, thereby increasing the dimensions of said matrix. The node then performs Gaussian elimination on the entire coding matrix, in order to determine if the newly received packet is linearly independent of the other packets in the decoding matrix. That is, the node attempt to decode the packets by using Gaussian elimination on the coding vectors. This paper considers network codes where the relaying node performs decoding.

# 3 Symbol Loss Probability and Delay

The source generate (or otherwise receive) symbols at a fixed rate. Whenever a new symbol is available the source will encode two packets and transmit them to the relay. The relay will then, when receiving a packet, perform Gaussian elimination, to establish if the newly received packet is linearly independent of the ones in storage. If the packet is linearly independent, the relay will recode two packets and transmit them to the sink. The sink decodes all the received packets, using Gaussian elimination. When using a dense code, the source does not transmit any packets before all the symbols of a block are received.

**Definition E.1 (Symbol Delay).** *When the sink decodes symbol $j$ and the source contains $i$ symbols, then the delay experienced by symbol $j$ is $i - j$. The delay experienced by undecodable symbols is not defined.* △

This definition of symbol delay does not take the channel latency into account. Furthermore, it is assumed that the channel is not saturated.

**Definition E.2 (Maximum Allowable Symbol Delay).** *Let $d$ be the maximum allowable symbol delay, then any symbol that experiences a delay larger than the maximum allowable symbol delay is considered lost.* △

**Definition E.3 (Symbol Loss Probability).** *Let $P_L(d; e, \mathcal{C}_1, \mathcal{C}_2, m)$ be the probability that a symbol is lost due to erasures. This probability is termed the symbol loss probability.* △

**Definition E.4 (Effective Symbol Loss Probability).**
*Let $P_E(d; e, \mathcal{C}_1, \mathcal{C}_2, m)$ be the probability that a symbol is lost either due to a large delay or erasures. This probability is termed the effective symbol loss probability.* △

In Section 4 the symbol loss probability and symbol delay, will be used to determine which code type (and which parameters) yields the lowest effective symbol loss probability given some maximum allowed delay and erasure probability of the channel.

## 3.1 Finite Recoder and Decoder Memory

We define $m \in \mathbb{N}$ as the memory factor that together with $v_j$ determines the number of packets the relay and sink may have in memory. That is, the relay may store at most $mv_2$ packets at any time. Note that only $m = 1$ is applicable to block codes, and we shall therefore only consider different values of $m$ for convolutional codes. The coding vectors of the at most $mv_2$ packets, in memory at the relay, are required to uphold the following

$$\min(\text{supp}(c_i)) - \min(\text{supp}(c_j)) < mv_2, 1 \leq j < i \leq mv_2, \tag{E.1}$$

where $c_i$ and $c_j$ are the $i$th and $j$th rows (coding vectors) of the decoding matrix. If this is not the case then the $j$th packet is removed from the decoding matrix. The condition on $i$ and $j$ come from the decoding process, which ensures that any two coding vectors do not have identical support: $\text{supp}(c_i) \neq \text{supp}(c_j), i \neq j$ and that the coding vectors are ordered:

$$\min(\text{supp}(c_i)) > \min(\text{supp}(c_j)), 1 \leq j < i \leq mv_2. \tag{E.2}$$

The sink has memory restrictions similar to those in (E.1) and (E.2), however the maximum number of packets in memory is $m \max(v_1, v_2)$ instead of $mv_2$. That is, the sink is allowed to hold at least the same number of packets as the relay, and potentially more. The reason for this is that the sink must be able to cope with the maximum code length, in order to properly decode the received packets.

The maximum code length from the source is $l_1 = v_1$ and if $\mathcal{C}_2$ is a block code then the maximum code length from the relay is $l_2 = v_2$. However, if $\mathcal{C}_2$ is a convolutional code then the relay is not limited to a code length of $v_2$. This is due to the continuous nature of convolutional codes.

**Lemma E.1.** *Let $\mathcal{C}_1$ be a rate $1/2$ code with constraint length $v_1$, let $\mathcal{C}_2$ be a $(2, 1, v_2 - 1)$ convolutional code, and let $m$ be the memory factor of the relay, then the maximum code length $l_2$ (from the relay) is*

$$l_2 = \min(v_1 v_2, mv_2 + v_1 - 1)$$

*Proof.* It is obvious that combining $v_2$ non-overlapping coding vectors, each with a maximum code length of $v_1$, results in a maximum code length of $v_1 v_2$. However, if the $v_2$ coding vectors are overlapping, then from (E.1) we see that they span across at most

$$mv_2 \geq 1 + \max_{1 \leq j < i \leq mv_2} (\min(\text{supp}(c_i)) - \min(\text{supp}(c_j)))$$

symbols. The last packet may then include contributions from another $v_1 - 1$ symbols, thus the maximum code length is the sum, i.e., $mv_2 + v_1 - 1$. □

**Definition E.5 (Symbol Delay Distribution).** *A symbol delay distribution is a probability mass function, which states that the probability that a decodable symbol will experience a delay of $d$ is $f_D(d; e, \mathcal{C}_1, \mathcal{C}_2, m) = Pr(D = d \mid e, \mathcal{C}_1, \mathcal{C}_2, m)$ given some erasure probability $e$, codes $\mathcal{C}_1$ and $\mathcal{C}_2$, and memory factor $m$, where $D \in \mathbb{N}_0$.* △

Let $e \in [0, 1]$ be the erasure probability, let $\mathcal{C}_1$ and $\mathcal{C}_2$ be erasure correcting codes, and $m \in \mathbb{N}$. Then the probability that a symbol experiences a delay less than or equal to $d$ is given by the cumulative distribution function of the delay, i.e.

$$Pr(D \leq d \mid e, \mathcal{C}_1, \mathcal{C}_2, m) = \sum_{i=0}^{d} f_D(i; e, \mathcal{C}_1, \mathcal{C}_2, m).$$

**Lemma E.2.** *Let $e \in [0, 1]$ be the erasure probability, let $\mathcal{C}_1$ and $\mathcal{C}_2$ be erasure correcting codes, and $m \in \mathbb{N}$. Then, given a symbol loss probability $P_L(e, \mathcal{C}_1, \mathcal{C}_2, m)$ and a maximum allowable delay $d$, then the effective symbol loss probability is given by:*

$$P_E(d; e, \mathcal{C}_1, \mathcal{C}_2, m) = 1 - Pr(D \leq d \mid e, \mathcal{C}_1, \mathcal{C}_2, m) \cdot (1 - P_L(e, \mathcal{C}_1, \mathcal{C}_2, m)). \quad \text{(E.3)}$$

*Proof.* $1 - P_L(e, \mathcal{C}_1, \mathcal{C}_2, m)$ is the probability of successful reception of a symbol. This probability is multiplied with the probability that a symbol experiences an allowable delay, which yield the probability that a symbol is both received and experiences an allowable delay. □

# 4 Simulation Method

Through simulations, we have obtained a set of empirical delay distributions parametrized by the tuple of $v_1$, $v_2$, and $e$ (and $m$ for convolutional codes). Where the constraint lengths took on the values $v_1, v_2 \in \{1, \ldots, 64\}$, erasure probabilities were $e \in \{0.05, 0.10, \ldots, 0.45\}$, and for the convolutional codes the memory factor was $m \in \{1, \ldots, 4\}$. In order to ensure that the simulations

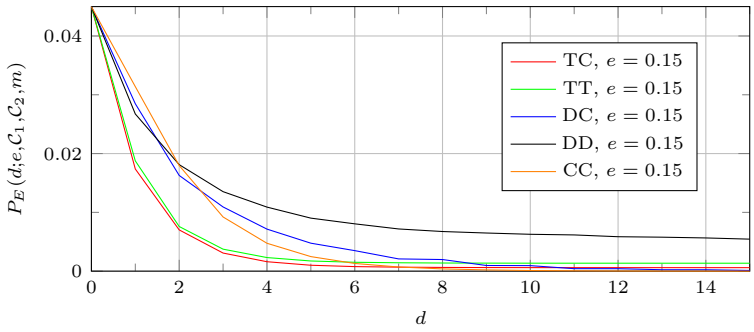



**Fig. E.2:** The effective symbol loss for all code types for $e = 0.15$.

use the same number of symbols for all three code types, we used $\left\lceil \frac{10^6}{v_1} \right\rceil v_1$ symbols in each simulation. Note that ideally we should have used $\text{lcm}(2, 3, \ldots, 64)$ symbols (where lcm refers to the least common multiple) to ensure that all simulations used the exact same number of symbols, however, this is not feasible[2]. The erasure pattern was identical for all code types. The coding coefficients were chosen at random from the finite field $\text{GF}(2^8)$, without using the zero element.

These delay distributions are then used to determine the effective symbol loss probability for each pair of $e$ and $d \in \{0, \ldots, 15\}$. The effective symbol loss probability is calculated for all delay distributions using (E.3). Then for each code type, the set of parameters yielding the lowest effective symbol loss probability is found for each pair of $e$ and $d$.

# 5 Discussion

Fig. E.2 show $P_E(d; e, \mathcal{C}_1, \mathcal{C}_2, m)$ for all of the code types for $e = 0.15$ and $d \in \{0, \ldots, 15\}$. From the figure it is clear that all code types, except DD, tend towards a similar limit, as $d$ increases. For $d > 7$ the CC code yields the lowest effective symbol loss, and for $d \leq 7$ it is the TC code.

The case of $e = 0.25$ and $e = 0.35$ is shown in Fig. E.3. From this figure we see that at the single point where $e = 0.35$ and $d = 1$ the DD code is the best performing code. At all other points on the figure the TC code out performs the other codes. Moreover, the TC code is significantly better than the DD code as $d$ increases. The same goes for the CC code when $e = 0.35$.

From a practical perspective, it may prove advantageous to use block codes to increase coding throughput, as shown in Paper A, and to simplify the implementation, by reusing the software implementation on both links. In this case,

[2] $\text{lcm}(2, 3, \ldots, 64) = 118226688410282226751136160 0 \approx 1.1823 \times 10^{27} \approx 1.9101 \times 2^{89}$

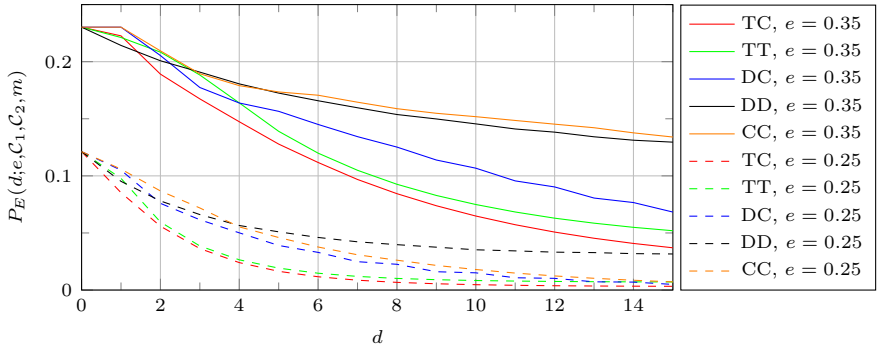



**Fig. E.3:** The effective symbol loss for all code types for $e = 0.25$ and $e = 0.35$.

**Table E.2:** Parameters for alle the code type, corrosponding to Figures E.2 and E.3. For TC, DC, and CC the parameters are: $v_1, v_2, m$. For TT and DD the parameter is $v_1$.

| e \ d | Code | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| 0.15 | TC | 1,1,1 | 2,60,1 | 3,64,3 | 13,64,4 | 5,64,3 | 22,64,2 | 22,64,2 | 22,64,2 |
| | DC | 1,1,1 | 2,2,1 | 2,2,2 | 3,3,3 | 3,3,3 | 47,3,3 | 33,4,2 | 33,4,2 |
| | CC | 1,1,1 | 2,1,1 | 3,2,1 | 4,3,1 | 5,4,1 | 6,5,1 | 35,15,3 | 30,42,4 |
| | TT | 1 | 61 | 62 | 62 | 62 | 62 | 62 | 62 |
| | DD | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 0.25 | TC | 1,1,1 | 2,60,1 | 3,60,1 | 4,64,1 | 5,63,1 | 44,63,4 | 44,64,3 | 12,64,4 |
| | DC | 1,1,1 | 2,2,1 | 2,2,3 | 2,2,3 | 3,3,3 | 4,3,4 | 4,4,2 | 8,4,4 |
| | CC | 1,1,1 | 2,1,1 | 2,1,3 | 3,2,3 | 3,2,3 | 3,2,3 | 4,3,3 | 4,3,3 |
| | TT | 1 | 7 | 60 | 63 | 63 | 64 | 64 | 64 |
| | DD | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 0.35 | TC | 1,1,1 | 2,5,1 | 2,6,2 | 2,7,4 | 3,59,1 | 4,62,1 | 5,64,1 | 5,64,1 |
| | DC | 1,1,1 | 1,1,1 | 2,2,2 | 2,2,2 | 2,2,4 | 2,2,4 | 3,3,3 | 3,3,3 |
| | CC | 1,1,1 | 1,1,1 | 2,1,2 | 2,1,2 | 2,1,4 | 2,1,4 | 2,1,4 | 3,1,3 |
| | TT | 1 | 2 | 5 | 10 | 61 | 61 | 61 | 61 |
| | DD | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

then the TT code is the one that overall yields the lowest effective symbol loss, quite significantly in some cases.

Table E.2 lists the parameters for each code type, which yielded the lowest effective symbol loss. For all codes the optimal constraint length is 1 when $d = 0$. As $d$ increases the parameters that yield the lowest effective symbol loss for the TT code is maximized. That is the optimal parameter for the TT code reaches (or gets very close to) the limits of our experiments. On the other hand, for the DD code, the optimal parameter is, quite naturally, $v_1 = v_2 = d + 1$, for $e < 0.45$. For the codes DC and CC, the constraint lengths are generally not more than 4 for moderate losses, i.e., $e > 0.15$.

## 5.1 Small Matrices and Low Delay

In [18] the authors of the present paper proposed the use of lower triangular Toeplitz coding matrices with static coding coefficients over small dimensions, e.g., $10 \times 10$, for low delay streaming. There are several benefits of these coding matrices. First, it allows for an increased coding throughput performance, by

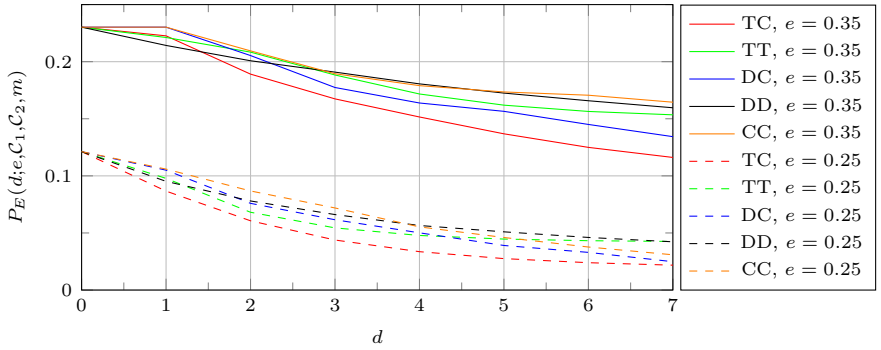



**Fig. E.4:** The effective symbol loss for all code types for $e = 0.25$ and $e = 0.35$, when $v \leq 10$.

carefully choosing the coding coefficients. Second, static coefficients provide predictable performance given that the coding coefficients are constant. In this scenario we showed in [16] that the TT code was superior for most sets of $d$ and $e$. However, by introducing the codes TC and DC we can see from Fig. E.4 that these two codes are able to reduce the symbol loss probability even further, as $d$ increases. It is also clear that the TC code is generally the code that produces the lowest symbol loss probability.

# 6 Conclusions

The findings in this paper are in line with those of [16]. That is, the TT code is the one that overall yields the lowest effective symbol loss when both $\mathcal{C}_1$ and $\mathcal{C}_2$ use the same structure (block or convolutional). But when we allow for block codes and convolutional codes to be used sequentially then the TC code is generally the code that produces the lowest symbol loss probability. We showed that the ranking of the codes does not significantly depend on the constraint length. This allows for the use of coding matrices of smaller dimensions, e.g., $10 \times 10$. The method described in this paper to obtain delay distributions and thereby also the effective symbol loss can easily be extended to other code types, code structures, and parameters.

# References

[1] S. Hahm, P. Kang, H. Bang, and H. Yeon, "Dynamic media buffer control scheme for seamless streaming in wireless local area networks," *IEEE Wireless Communications and Networking Conference Workshops*, pp. 109–114, 2016.

[2] J. Hansen, J. Krigslund, D. E. Lucani, and F. H. P. Fitzek, "Sub-Transport Layer Coding: A Simple Network Coding Shim for IP Traffic," *IEEE 80th Vehicular Technology Conference*, pp. 1–5, 2014.

[3] H. Schulzrinne, A. Rao, R. Lanphier, M. Westerlund, and M. Stiemerling, "Real-Time Streaming Protocol Version 2.0," RFC 7826 (Proposed Standard), RFC Editor, pp. 1–318, December 2016.

[4] M. Tuexen, I. Ruengeler, and R. Stewart, "SACK-IMMEDIATELY Extension for the Stream Control Transmission Protocol," RFC 7053 (Proposed Standard), RFC Editor, pp. 1–8, November 2013.

[5] M. Wang and B. Li, "Network Coding in Live Peer-to-Peer Streaming," *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1554–1567, December 2007.

[6] H. Seferoglu and A. Markopoulou, "Opportunistic network coding for video streaming over wireless," *Packet Video 2007*, pp. 191–200, November 2007.

[7] C. Gkantsidis and P. R. Rodriguez, "Network coding for large scale content distribution," *IEEE Computer and Communications Societies*, vol. 4, pp. 2235–2245, March 2005.

[8] P. Patil, A. Badr, A. Khisti, and W. T. Tan, "Delay-optimal streaming codes under source-channel rate mismatch," *Asilomar Conference on Signals, Systems and Computers*, pp. 2094–2099, November 2013.

[9] C. E. Luna, Y. Eisenberg, R. Berry, T. N. Pappas, and A. K. Katsaggelos, "Joint source coding and data rate adaptation for energy efficient wireless video streaming," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 10, pp. 1710–1720, December 2003.

[10] K.-W. Lee, R. Puri, T. eun Kim, K. Ramchandran, and V. Bharghavan, "An integrated source coding and congestion control framework for video streaming in the Internet," *IEEE International Conference on Computer Communications (INFOCOM)*, vol. 2, pp. 747–756, 2000.

[11] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, April 1950.

[12] P. Elias, "Coding for noisy channels," *IRE Convention Records Part 4*, pp. 37–46, 1955.

[13] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[14] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," *IEEE International Symposium on Information Theory*, June 2003.

[15] J. Hansen, J. Krigslund, D. E. Lucani, and F. H. P. Fitzek, "Bridging inter-flow and intra-flow network coding for video applications: Testbed description and performance evaluation," *IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks*, pp. 7–12, September 2013.

[16] J. Hansen, J. Østergaard, J. Kudahl, and J. Madsen, "When are Erasure Correcting Block Codes Better than Convolutional Codes in a Multi-hop Network?" *11th International Conference on Signal Processing and Communication Systems*, December 2017.

[17] H. Gluesing-Luerssen, J. Rosenthal, and R. Smarandache, "Strongly-MDS convolutional codes," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 584–598, February 2006.

[18] J. Hansen, J. Østergaard, J. Kudahl, and J. H. Madsen, "Superregular lower triangular toeplitz matrices for low delay wireless streaming," *IEEE Transactions on Communications*, vol. 65, no. 9, pp. 4027–4038, September 2017.

ISSN (online): 2446-1628
ISBN (online): 978-87-7210-121-7

AALBORG UNIVERSITY PRESS