Performance Assessment of Fountain-coded Schemes for Progressive Packet Recovery

Andrew Jones

School of Computing and Communications

Lancaster University

This dissertation is submitted for the degree of

Master of Science by Research

November 2014

I dedicate this thesis to:
My father, Colin Jones
My mother, Sarah Jones
And my considerably better half, Alex Earley

Declaration

Andrew Jones
November 2014

Acknowledgements

Firstly, I would like to thank my supervisor Dr. Ioannis Chatzigeorgiou for his constant support and encouragement throughout this project. I cannot thank him enough for putting up with my perpetual pestering and my “short” meetings, which usually ended up taking a couple of hours. Once more, I offer my sincere apologies for accidentally omitting this acknowledgement page in my third year project report!

I would also like to extend my thanks to the Engineering and Physical Sciences Research Council who, via the funding of the R2D2: Network error control for **R**apid and **R**eliable **D**ata **D**elivery research project, allowed me to experience my first academic conference.

My wholehearted thanks to those in Room B38, in particular Dr. Andrea Tassi and Klen Ĉopiĉ Pucihar, who not only provided an enjoyable and friendly working environment, but also served to remind me that there is more to life than constant academic research with fruitful discussions during our regular *café* sessions.

Finally, my heartfelt thanks to my parents, Colin and Sarah, and my better half Alex, for their love and support over the years and also their trust in my decisions - “What! You want to do *another* Masters?!”.

Abstract

Fountain Codes are a more efficient method of broadcasting identical messages to multiple recipients when compared with more conventional multicast schemes such as Automatic-Repeat-reQuest. This is because Fountain Codes transmit random combinations of the source packets, and hence, do away with the requirement of a costly feedback channel. This feedback channel is used by recipients to request that the transmitter resends the specific source packets that they have not yet recovered.

Usually, Fountain Coded schemes focus on the recovery of the *entire* source message with minimal overhead. This, of course, requires a recipient to receive a number of coded packets that it is at least equal to the number of original source packets. However, if a *multimedia* file was being broadcast using a fountain code, what if it could be guaranteed that the recipient would receive an ordered portion, for example the first half, of the source message before receiving the full amount of coded packets needed to recover the *entire* source message. The user could then start playing the media earlier, so long as the rest of the message is recovered without incurring an unrealistic delay. It is this proposition that has motivated this work and has also culminated in a published conference paper [1].

During this thesis four schemes are contrasted, three exhibiting fountain coding and a benchmark scheme. Theoretical expressions for the first metric, the probability of recovering *every* source packet, are derived for the majority of the schemes. A second metric is then introduced, the probability of recovering a *portion* of the source message, and additional theoretical expressions are derived to take this into account. In the preparation of the MATLAB platform used to obtain simulation results, three different branches of decoding algorithm are examined and then adapted, in order to find the most suitable candidate. Simulation results for each of the encoding schemes are then presented and discussed.

Lastly, the concept of a *layered* source message is introduced, where some source packets are more important than others, and some of the schemes present in literature that attempt to tailor generic fountain codes for use with scalable video encoders. To examine the progressive and overall performance of these "layered" schemes, and to see if their progressive performance can be enhanced at all, the original encoding schemes are modified and simulation results are presented and discussed.

Contents

List of Figures

List of Tables

Code Listings

Chapter 1

Introduction

The broadcast of identical messages to multiple recipients within a wireless medium was, historically, an extremely inefficient process. Ubiquitous repeating protocols such as Automatic-Repeat-reQuest (ARQ) led to intolerably large delays and redundant transmissions. In an ARQ scheme, the source message is initially broadcast by the transmitter in an ordered fashion, and once this has been accomplished, receivers then request the retransmission of the specific source packets that they have not yet recovered because of the lossy nature of the wireless channel [5].

Figure 1.1 shows an ARQ-style broadcast of a three packet source message from transmitter S to receivers R_1, R_2 and R_3. R_1 has successfully recovered s_1 and s_2, R_2 has successfully recovered s_2 and s_3 and R_3 has only recovered s_2. In the next stage of the ARQ strategy, R_1 would request that S retransmits s_3, R_2 would request the retransmission of s_1 and R_3 would request the transmission of both s_1 *and* s_3. From the redundant retransmission requests of R_3, it is evident, from this example, that this phase of the ARQ scheme is inefficient when utilised for multicast.

The wireless medium exhibits a broadcast effect, without which wireless multicast could not occur, that means that anything transmitted by S can also be heard by R_1, R_2 *and* R_3, and at this point in the ARQ scheme, this means that these two recipients might receive redundant packets that they have already recovered [6].

To mitigate the inefficiencies of conventional ARQ schemes when used in wireless multicast scenarios, in 1998 Byers *et al.* [7] introduced the concept of fountain coding. When fountain codes are employed, source packets are not transmitted in an ordered fashion, as they were in ARQ, but coded packets are transmitted instead. These coded packets are linear combinations (XORs) of a randomly selected amount of source packets.

So, instead of a receiver looking to recover specific packets, such as s_1, s_2 and

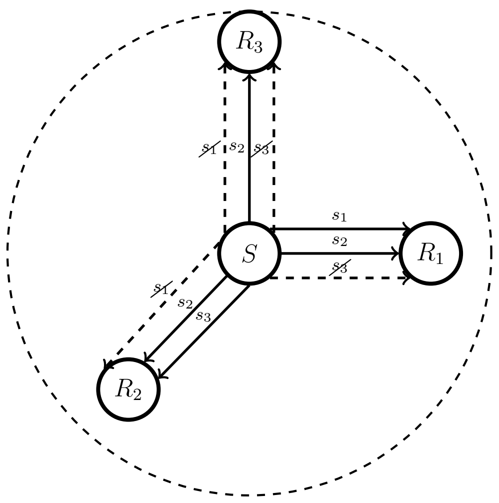


Fig. 1.1 A toy example of an ARQ-style broadcast of a three packet source message. The dashed circle represents the area within which transmissions from S can be heard.

s_3, the receiver only wishes to recover at least three coded packets. Each of these coded packets can be considered as a simultaneous equation, with each equation having three unknowns. From elementary algebra, it can be concluded that at least three coded packets must be recovered to have any chance of recovering the three source packets.

Fountain codes as they were initially proposed, now known as Random Linear Fountain Codes, had a high encoding and decoding complexity because of the sheer amount of XOR row operations that were required to produce and then decode each coded packet. This stemmed from the fact that there were no restrictions on the number of source packets that could be included in each coded packet.

To reduce the computational complexity of these original codes, Luby proposed a modification in 2002 [8] called Luby-Transform codes, where the number of packets that are included in each coded packet is not uniformly random, but is based on a probability distribution known as a *degree distribution.* In his paper, Luby also proposed a methodology for defining an efficient degree distribution, that can be tailored to different system requirements, known as the Robust Soliton Distribution. Random Linear Fountain Codes and Luby-Transform codes will be further investigated in Chapter 2.

Luby-Transform codes succeeded in lowering the complexity of Random Linear Fountain Codes, but they still exhibited an encoding and decoding complexity that scales with $\log_e(K)$, where K is the number of source packets in the message. However, this issue was quickly remedied with Shokrollahi's introduction of Raptor codes in 2006 [9], which exhibit *linear* time encoding and decoding.

Because of their low complexity and overhead, when compared to other codes used for multimedia multicast, Raptor codes have found application within recent standards such as 3GPP [10] and DVB-H [11].

In every fountain coded scheme mentioned above, the focus has always been the recovery of the entire source message with minimal overhead. That, of course, requires a recipient to have received at least K coded packets. However, if a *multimedia* file was being broadcast using a fountain code what if it could be guaranteed that the recipient would receive an ordered portion, for example the first half, of the source message. The user could then start playing the media, as long as the rest of the message is recovered without incurring an unrealistic delay, before receiving the full K coded packets. It is this proposition that has motivated this work and has also culminated in a published conference paper [1], which can be seen in Appendix A, and a second conference paper [12] that is currently under review, which can be seen in Appendix B.

1.1 Objectives

The objectives of this work can be summarised as follows.

- Perform a literature review into known fountain coding and decoding techniques.
- Become familiar with stochastic methods for evaluating the probability of successful decoding at a receiving end.
- Produce an efficient simulation platform in MATLAB and validate its accuracy using known results.
- Develop or adapt existing decoding mechanisms to progressively (on-the-fly) recover information as more packets are recovered at a receiver.
- Derive theoretical expressions for the progressive and overall performance of these mechanisms.
- Compare theoretical results and simulation results for on-the-fly packet recovery.
- Introduce the concept of a layered source message and the adapted fountain coding schemes that are employed to broadcast such messages.
- Adapt the simulation platform to allow simulations for layered source information and examine and discuss simulation results for layered scenarios.

The thesis is structured as follows to first provide background knowledge, and then to build upon and extend fundamental encoding schemes, in order to achieve the project objectives mentioned above. In Chapter 3 the four encoding schemes are introduced, three exhibiting fountain coding and a benchmark scheme. Theoretical expressions for the first metric, the probability of recovering *every* source packet, are then derived for the majority of the schemes. A second metric, the probability of recovering a *portion* of the source message, is then introduced and additional theoretical expressions are derived.

As mentioned above, fountain coded packets can be viewed as simultaneous equations so it is not particularly surprising that many fountain decoding techniques are built around the Gaussian Elimination process. In Chapter 4, three different branches of decoding algorithm are examined and adapted, in order to find the most suitable candidate for use in the MATLAB simulation platform.

Chapter 5 then presents and discusses simulation results for each of the encoding schemes mentioned in Chapter 3.

Recently, the Joint Video Team of the ITU-T VCEG and the ISO/IEC MPEG standardised a Scalable Video Coding (SVC) extension of the H.264/AVC standard [13]. In previous chapters, the performance of *layerless* encoding schemes has been examined, where every source packet has *identical* importance, to see how well receivers can progressively recover the media message. However, a video encoder that utilises the SVC extension produces *multiple* layers, each with a different level of importance.

In Chapter 6, the concept of a *layered* source message is introduced, as well as some of the schemes already present in literature that attempt to tailor generic fountain codes for use with scalable video encoders. To examine the progressive and overall performance of these "layered" schemes, and to see if their progressive performance can be enhanced at all, modifications to Chapter 3's encoding schemes are introduced. Simulation results for the modified encoding schemes are then presented and discussed.

This thesis is then summarised in Chapter 7, and the main contributions of this work are listed. Possible areas for future research are also highlighted.

Chapter 2

Fundamentals of Fountain Coding

In this chapter, the theoretical performance of the Random Linear Fountain Code (RLFC) is examined. A RLFC is a method of disseminating identical source messages to multiple users, that was initially presented by Byers *et al.* [7]. The analysis will first allow and then remove the possibility of a null coded packet. Finally, the encoding and decoding process for the first practical implementation of the Fountain Code technique, the Luby Transform code [8] will be touched on.

This introduction to generic Fountain Coding principles, and the method of their investigation, will be built upon in later chapters.

2.1 Encoding Process of a RLFC

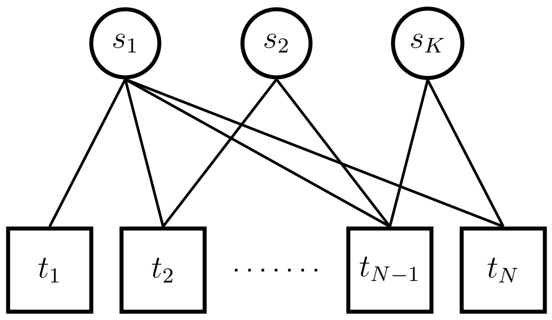
If a Random Linear Fountain Code was to encode N packets over a message comprised of K source packets $\{s_1, s_2, \ldots, s_K\}$, the encoder would construct a coded packet t_n at time step n, where $\{n = 1, \ldots, N\}$, from the linear combination or, in other words, the bitwise sum of source packets as follows [14]

$$t_n = \sum_{i=1}^{K} g_{n,i}\, s_i \tag{2.1}$$

where $g_{n,i}$ is a binary coding coefficient selected uniformly at random.

Figure 2.1 displays the construction of each coded packet in a scenario with $K = 3$ source packets. To give a specific example, coded packet t_2, which has been generated with random coding vector $G_2 = [1\ 1\ 0]$, would contain the bitwise sum of s_1 and s_2 ($t_2 = s_1 \oplus s_2$).

Once the coded packets have been generated, they are transmitted to a receiver over a potentially lossy channel. This channel is modelled as a packet erasure channel (PEC), where the probability of not recovering a particular transmitted packet is p. This behaviour can be seen in Figure 2.2 for a similar scenario to that



(a)

	s_1	s_2	s_K
G_1	1	0	0
G_2	1	1	0
	$\vdots$		
G_{N-1}	1	1	1
G_N	1	0	1

(b)

Fig. 2.1 The interconnects between each source packet and each coded packet (a), given a specific encoding matrix (b).

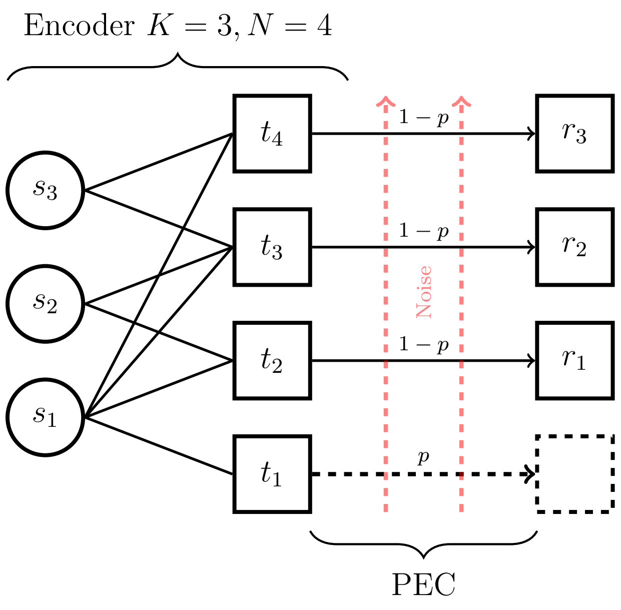


Fig. 2.2 The construction and transmission of the coded packets over the PEC.

in Figure 2.1 but with $N = 4$ transmitted coded packets. Packets r_1, r_2 and r_3 represent the $i = 3$ transmitted coded packets that have actually been successfully recovered from the lossy channel by the receiver.

2.2 Decoding Process of a RLFC

In order to successfully recover all of the K source packets, at least K linearly independent coded packets must be recovered. Another way of phrasing this requirement, is that the coding vectors of the received packets must form a matrix of rank K.

The row-echelon form of the received encoding matrix can be found by applying Gaussian Elimination [15], the original matrix's rank can then be obtained by inspecting the number of non-zero rows within the echelon form.

```
function [ cod_matrix ] = GaussElim( cod_matrix )

[num_coded, num_source] = size(cod_matrix);

% For each position on the diagonal
for arg_count = 1 : num_source

    % As a row-echelon form matrix needs to be formed
    % check if the current diagonal element
    % is equal to 0
    if( cod_matrix(arg_count,arg_count) == 0 )

        % As this element needs to be 1, check the other
        % encoding vectors down the diagonal, and swap if
        % they have a 1 in the element that is currently
        % being examined
        for row_count = arg_count : num_coded
            if(cod_matrix(row_count,arg_count) ~= 0 )
                temp = cod_matrix(arg_count, :);
                cod_matrix(arg_count, :) = ...
                    cod_matrix(row_count,:);
                cod_matrix(row_count,:) = temp;
            end
        end
    end

    % Now that there is a 1 in the correct position on the
    % diagonal, XOR the current encoding vector with every
    % other encoding vector to make sure that the current
    % vector is the only one with this element on the
    % diagonal
    for row_count = 1 : num_coded
        if( row_count ~= arg_count )
            if( cod_matrix(row_count,arg_count) )
                cod_matrix(row_count,:) = ...
                    xor(cod_matrix(row_count,:), ...
                    cod_matrix(arg_count,:));
            end
        end
    end

end
```

Listing 2.1 Simple example of Binary Gaussian Elimination algorithm used to obtain the row-echelon form of the received encoding matrix.

```
function [ recoverable_packets ] = Back_Sub( cod_matrix )

  % Initially, no source packets have been recovered
  recoverable_packets = zeros(1, num_source);

  % Counting upwards from the furthest element down
  % the diagonal, as the encoding matrix has been
  % ordered by the GE process
  for i = num_source : -1 : 1
      % If there is a single non-null element in
      % the current coding vector, then this element
      % can be recovered
      if sum(cod_matrix(i,:)) == 1
          % Note that it has been recovered
          recoverable_packets(cod_matrix(i,:) == 1) = 1;
          % And remove the recovered element from
          % any other coding vectors
          cod_matrix(:,cod_matrix(i,:) == 1) = 0;
      end
  end

end
```

Listing 2.2 MATLAB implementation of the simple back-substitution phase that would accompany the GE process in Listing 2.1.

For example, to obtain the row-echelon form of the encoding matrix received in the scenario given in Figure 2.2, follow the algorithm presented in Listing 2.1. The resultant matrix transformations would be those in Figure 2.3.

Towards the end of Figure 2.3, the row-echelon matrix has $K = 3$ non-zero rows. This implies that the received encoded matrix is not rank-deficient and that all of the source packets can be recovered immediately. In this case there is no need to wait for any further innovative packets to arrive. The phrase “innovative packet” is used to define a coded packet that is linearly independent of the received coded packets or, in other words, is *not* a bitwise sum of the binary coded packets that have already been received.

To pick out which source packets are recoverable from a rank-deficient matrix of received coding vectors, instead of simply checking an encoding matrix for full rank, a simple secondary back-substitution stage could be appended to Listing 2.1. This back-substitution stage would take the resulting row-echelon form matrix, once the GE process had completed, and would attempt to transform it into a *reduced* row-echelon form.

A MATLAB implementation of this process can be seen in Listing 2.2.

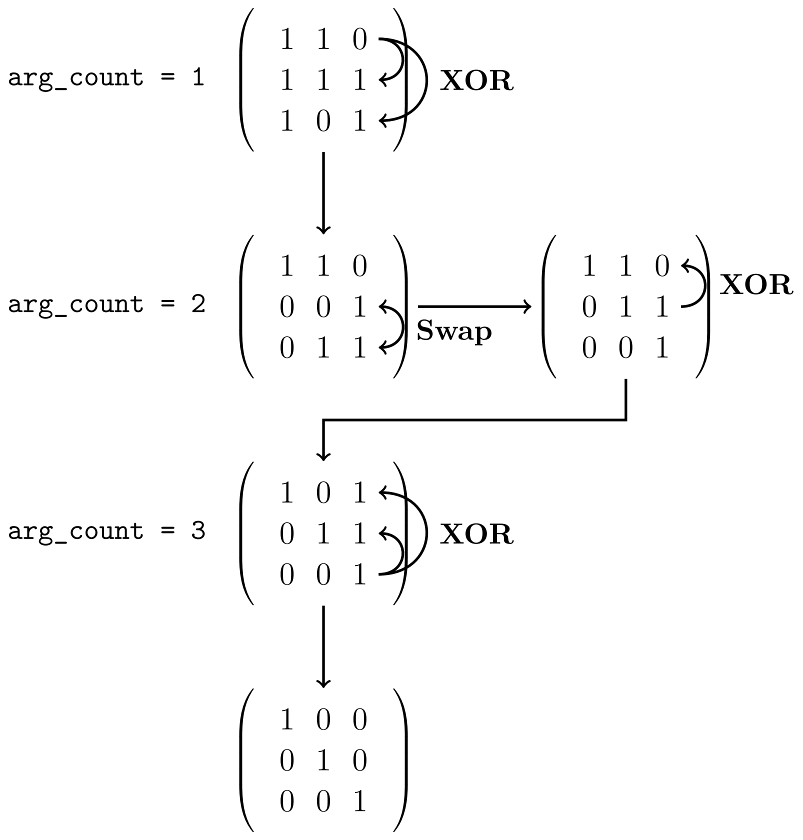


Fig. 2.3 An example showing the Gaussian Elimination algorithm presented in Listing 2.1 when applied to the received coding matrix in Figure 2.2.

2.3 Theoretical Performance of a RLFC

As random combinations of source packets are being transmitted, the theoretical performance of Random Linear Fountain Codes is not all-encompassed by the probability of simply receiving K transmitted packets. The probability that the received "simultaneous equations" are solvable needs to be taken into account. In other words, is the matrix of the received encoding vectors invertible? Are the $N \geq K$ received encoding vectors *linearly independent*?

This probability of the received packets being linearly independent will now be investigated. The possibility of randomly generating a coded packet that is not linked with any source packets, known as a null coded packet, will be initially included in the probability analysis for simplicity but then it will be removed.

Once expressions have been obtained for the probability that K of the i received packets are linearly independent, this investigation will be concluded by including the probability of actually receiving the i packets in the first place.

2.3.1 Including the possibility of a null coded packet

A toy example, of a RLFC encoder generating coded packets over $K = 3$ source packets, will initially be considered. It is assumed that the receiver has managed to obtain $i = 3$ fountain coded packets from the lossy channel.

Afterwards, generic expressions for any value of $i = K$ and then for any value of $i \geq K$ will be found. However, at this point, the next step is to investigate the probability that the $i = 3$ received packets are linearly independent.

To obtain this probability of the linear independence of $i = 3$ received fountain coded packets, that were encoded over $K = 3$ source packets, the first step is to consider all of the possible combinations of source packets that could be present in a coded packet. These combinations are shown as columns in the matrix below:

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

As the null coded packet does not contain any information, it should not be selected when choosing the initial packet. The probability of initially *not* selecting the null coded packet is $1 - \frac{1}{2^K} = 1 - \frac{1}{8} = \frac{7}{8}$.

Now that a non-null packet has been selected, the second received coded packet should also be different to the null packet but also to the first coded packet that has been selected. The probability of not selecting the null packet or the initial packet again is $1 - \frac{2}{2^K} = 1 - \frac{2}{8} = \frac{6}{8}$.

Two of the three coded packets have now been considered. The third packet must be linearly independent to the previous two packets. This probability of linear independence is $1 - \frac{4}{2^K} = 1 - \frac{4}{8} = \frac{4}{8}$.

The combined probability of receiving three *linearly independent* coded packets is $\frac{7}{8} \cdot \frac{6}{8} \cdot \frac{4}{8} = \frac{21}{64}$.

This process can be generalised as follows for $i = K$:

$$\Pr_{\mathrm{LIN}}(K, i = K) = \prod_{a=0}^{K-1} \left(1 - \frac{2^a}{2^K}\right). \tag{2.2}$$

If $\frac{2^a}{2^K}$ is simplified, $\frac{2^a}{2^K} = 2^a \cdot 2^{-K} = 2^{a-K} = \frac{1}{2^{K-a}}$, the expression supplied by MacKay in [14] is found.

$$\Pr_{\mathrm{LIN}}(K, i = K) = \prod_{a=0}^{K-1} \left(1 - \frac{1}{2^{K-a}}\right). \tag{2.3}$$

If more than K coded packets are received ($i \geq K$), K of the received coded

	Packet 1	**Packet 2**	**Packet 3**	**Packet 4**
Combination 1	$\frac{1}{8}$	$\frac{7}{8}$	$\frac{6}{8}$	$\frac{4}{8}$
2	$\frac{7}{8}$	$\frac{2}{8}$	$\frac{6}{8}$	$\frac{4}{8}$
3	$\frac{7}{8}$	$\frac{6}{8}$	$\frac{4}{8}$	$\frac{4}{8}$
4	$\frac{7}{8}$	$\frac{6}{8}$	$\frac{4}{8}$	$\frac{8}{8}$

Table 2.1 Possible combinations of the $i = 4$ received coded packets that allows the recovery of the $K = 3$ source packets.

packets must be linearly independent, while $i - K$ of the received packets will be linearly dependent.

For example, if $K = 3$ source packets were encoded over and the receiver obtained $i = 4$ coded packets, there are $\binom{i}{i-K} = \binom{4}{1} = 4$ different combinations of received packets that allow the recovery of the K source packets. These combinations can be visualised as in Table 2.1.

The **dark grey** fractions indicate packets which *are* linearly dependent. For example, the third row's packet combination implies that the first packet was not the null packet and that the second was linearly independent to the first. However, the third received packet *was* linearly dependent to the first and second, this event occurs with a probability of $\frac{4}{2^K}$ which is the complement of the probability of the third packet being linearly *independent*. After receiving a linearly dependent packet, the probability of the final packet being linearly independent to the first and second is still $1 - \frac{4}{2^K}$.

The set of light grey probabilities within each case are repeated throughout each row and can be defined by (2.2). If only the *differences* between each row are considered, the following set is obtained $\left\{\frac{1}{8}, \frac{2}{8}, \frac{4}{8}, \frac{8}{8}\right\}$. This set can be described for $i = K + 1$ by $\sum_{d_1=0}^{K} \frac{2^{d_1}}{2^K}$.

Hence, the probability of the linear independence of $i = K + 1$ received coded packets can be described as:

$$\Pr_{\text{LIN}}(K, i = K + 1) = \prod_{a=0}^{K-1} \left(1 - \frac{1}{2^{K-a}}\right) \cdot \sum_{d_1=0}^{K} \frac{2^{d_1}}{2^K}. \tag{2.4}$$

To obtain a generic expression for *any* value of $i \geq K$, a scenario where $i = 5$ coded packets have been received, which have been encoded over $K = 3$ source packets, is considered. Following identical logic to the investigation of the previous case, the $K = 3$ source packets can be recovered if $K = 3$ of the $i = 5$ received coded packets are linearly independent, the $(i - K) = 2$ remaining received coded packets will be linearly dependent.

Packet 1	**Packet 2**	**Packet 3**	**Packet 4**	**Packet 5**
$\frac{1}{8}$	$\frac{1}{8}$	$\frac{7}{8}$	$\frac{6}{8}$	$\frac{4}{8}$
$\frac{1}{8}$	$\frac{7}{8}$	$\frac{2}{8}$	$\frac{6}{8}$	$\frac{4}{8}$
$\frac{1}{8}$	$\frac{7}{8}$	$\frac{6}{8}$	$\frac{4}{8}$	$\frac{4}{8}$
$\frac{1}{8}$	$\frac{7}{8}$	$\frac{6}{8}$	$\frac{4}{8}$	$\frac{8}{8}$
$\frac{7}{8}$	$\frac{2}{8}$	$\frac{2}{8}$	$\frac{6}{8}$	$\frac{4}{8}$
$\frac{7}{8}$	$\frac{2}{8}$	$\frac{6}{8}$	$\frac{4}{8}$	$\frac{4}{8}$
$\frac{7}{8}$	$\frac{2}{8}$	$\frac{6}{8}$	$\frac{4}{8}$	$\frac{8}{8}$
$\frac{7}{8}$	$\frac{6}{8}$	$\frac{4}{8}$	$\frac{4}{8}$	$\frac{4}{8}$
$\frac{7}{8}$	$\frac{6}{8}$	$\frac{4}{8}$	$\frac{4}{8}$	$\frac{8}{8}$
$\frac{7}{8}$	$\frac{6}{8}$	$\frac{4}{8}$	$\frac{8}{8}$	$\frac{8}{8}$

Table 2.2 Possible combinations of the $i = 5$ received packets that allows the recovery of the $K = 3$ source packets.

These $\binom{5}{2} = 10$ combinations can be visualised as in Table 2.2, where the grey fractions now indicate one of the linearly dependent packets:

As with the previous example, there are repeating sets of probabilities defined by (2.2) within each row. It is now straightforward, by fixing the position of one of the dependent packets and iterating the position of the other, to extend (2.4) to take into account the probabilities introduced by an extra linearly dependent packet:

$$\Pr_{\text{LIN}}(K, i = K+2) = \prod_{a=0}^{K-1} \left(1 - \frac{1}{2^{K-a}}\right) \cdot \sum_{d_1=0}^{K} \frac{2^{d_1}}{2^K} \sum_{d_2=d_1}^{K} \frac{2^{d_2}}{2^K}. \tag{2.5}$$

By following the same reasoning for any value of $i \geq K$, in concurrence with [16], it is evident that:

$$\Pr_{\text{LIN}}(K, i) = \prod_{a=0}^{K-1} \left(1 - \frac{1}{2^{K-a}}\right) \cdot \sum_{d_1=0}^{K} \frac{2^{d_1}}{2^K} \sum_{d_2=d_1}^{K} \frac{2^{d_2}}{2^K} \cdots \sum_{d_{(i-K)}=d_{(i-K)-1}}^{K} \frac{2^{d_{(i-K)}}}{2^K}. \tag{2.6}$$

2.3.2 Without the possibility of a null encoded packet

In practice, as it contains no information, the possibility of a null encoded packet is removed. To revise the expression of $\mathrm{Pr}_{\mathrm{LIN}}$ to accommodate this, in a similar vein to the prior investigation, the first step is to consider all of the possible combinations of $K = 3$ source packets that could be present in a coded packet. These combinations are shown, as columns, in the matrix below:

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

As the possibility of an all-zero encoded packet has been removed, the total number of combinations is now $(2^K - 1) = 7$.

Initially, if any packet from the options above was selected, the probability of the second packet being linearly independent is $1 - \frac{1}{7} = 1 - \frac{1}{2^K-1} = \frac{6}{7}$. In other words, the probability of not selecting the first packet again.

The probability that the third packet is linearly independent to the first two packets is $1 - \frac{3}{7} = \frac{4}{7}$. The combined probability is $\frac{4}{7} \cdot \frac{6}{7} = \frac{24}{49}$.

Again, this process can be generalised as follows for $i = K$:

$$\mathrm{Pr}_{\mathrm{LIN}}(K, i = K) = \prod_{a=1}^{K-1} \left(1 - \frac{2^a - 1}{2^K - 1}\right). \tag{2.7}$$

Following the same process that resulted in (2.6), but starting with (2.7) instead of (2.2), $\mathrm{Pr}_{\mathrm{LIN}}$ can be redefined as:

$$\begin{aligned} \mathrm{Pr}_{\mathrm{LIN}}(K, i) = \prod_{a=1}^{K-1} \left(1 - \frac{2^a - 1}{2^K - 1}\right) \cdot \\ \cdot \sum_{d_1=0}^{K} \frac{2^{d_1} - 1}{2^K - 1} \sum_{d_2=d_1}^{K} \frac{2^{d_2} - 1}{2^K - 1} \cdots \sum_{d_{(i-K)}=d_{(i-K)-1}}^{K} \frac{2^{d_{(i-K)}} - 1}{2^K - 1}. \end{aligned} \tag{2.8}$$

2.3.3 Implementation of the Expression

Although (2.8) provides an *exact* expression for the probability that the i received packets are linearly independent, unfortunately, the multiplication of the nested sums does not lend itself to timely computation for $K \geq 20$. To overcome this issue, an approximation was introduced.

If (2.7) is revisited, an incomplete way of extending the expression for any

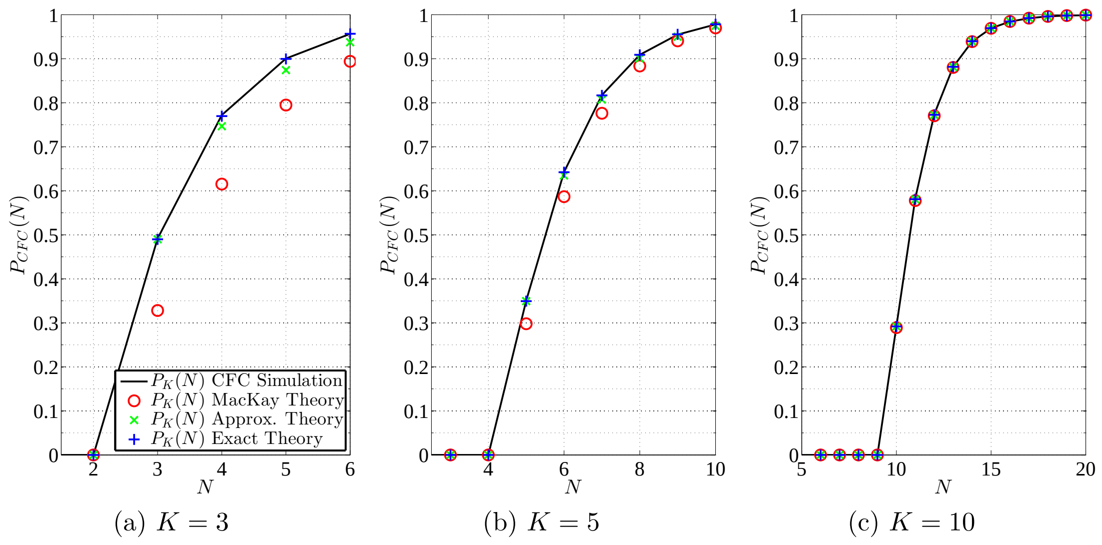


Fig. 2.4 Plot showing the accuracy of the three theoretical expressions for $\mathrm{Pr}_{\mathrm{LIN}}$ with no possibility of a null coded packet.

number of received packets would be:

$$\mathrm{Pr}_{\mathrm{LIN}}(K,i) \approx \prod_{a=1}^{K-1} \left(1 - \frac{2^a - 1}{2^i - 1}\right). \tag{2.9}$$

Equation (2.9) is the result of simply acknowledging that if a matrix has K linearly independent *rows* it must have K linearly independent *columns* [17]. The approximation is not exact because even though there is no possibility of an all-zero *column*, there is still a possibility of an all-zero *row*.

In [14], MacKay applies this row/column switch logic to produce an *exact* expression for the probability of i received coded packets being linearly independent. However, as the possibility of an all-zero coded packet is *always* considered, MacKay's theory is only exact for larger values of K, when the probability of generating a null coded packet gets very small.

$$\mathrm{Pr}_{\mathrm{LIN}}(K,i) \approx \prod_{a=0}^{K-1} \left(1 - \frac{1}{2^{i-a}}\right). \tag{2.10}$$

Figure 2.4 compares a simulation of a non-systematic conventional fountain code against the theoretical results given by (2.8), (2.9) and (2.10). It is evident that, although not strictly exact for $K < 10$, the proposed approach provides an acceptable approximation to the simulation results, considering the reduction in computational complexity.

2.3.4 End-to-end probability of packet recovery

Sections 2.3.1 and 2.3.2 examined the probability of overall packet recovery *given* that i coded packets have been received. This concluding section takes into account the lossy channel's erasure probability p, and the fact that if $N \geq K$ coded packets are transmitted, there are multiple values of $N \geq i \geq K$ that will allow the recovery of the K source packets.

If, as in the system model in Figure 2.2, p represents the probability of not receiving a particular packet, then $(1-p)$ becomes the probability of successfully recovering a particular packet. It follows that the probability of receiving three packets is $(1-p)^3$.

The probability of one instance of recovering exactly i of the N coded packets transmitted using RLFC is $(1-p)^i p^{N-i}$. There are $\binom{N}{i}$ possible ways of choosing the i received packets from the N transmitted packets, where $\binom{N}{i}$ denotes the binomial coefficient [18]

$$\binom{N}{i} \equiv \frac{N!}{(N-i)!\, i!} \tag{2.11}$$

where $z!$ denotes a factorial.

It follows that the complete probability of recovering exactly i of the N source packets transmitted is the binomial distribution

$$\binom{N}{i}(1-p)^i p^{N-i}. \tag{2.12}$$

Next, (2.12) is summed over all of the possible valid values of i that allows the recovery of at least K coded packets:

$$\sum_{i=K}^{N} \binom{N}{i}(1-p)^i p^{N-i} \tag{2.13}$$

Of course, to actually recover each of the source packets, the probability that there are K linearly independent packets within the i received packets needs to be included:

$$\mathrm{P}_K(N) = \sum_{i=K}^{N} \binom{N}{i}(1-p)^i p^{N-i}\, \mathrm{Pr}_{\mathrm{LIN}}(K,i) \tag{2.14}$$

where the definition of $\mathrm{Pr}_{\mathrm{LIN}}(K,i)$ can either be (2.6) or (2.8).

2.4 Luby Transform Codes

Luby Transform (LT) [8] codes were the first practical implementation of the fountain coding principles introduced in the previous section. They attempted to

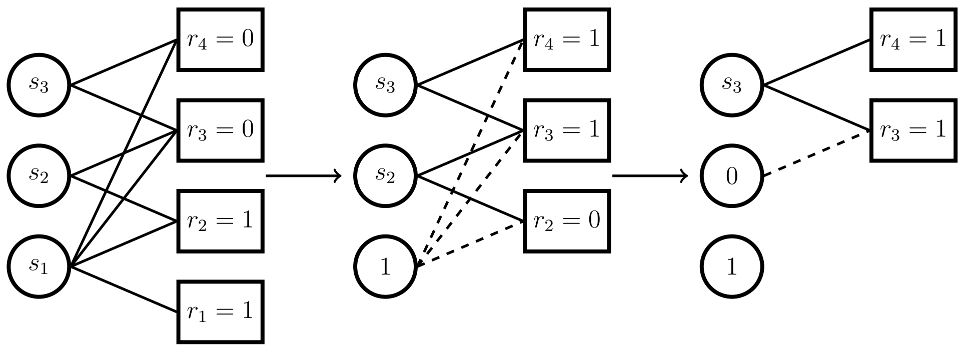


Fig. 2.5 An example of the Iterative Belief Propagation algorithm when applied to the coded packets from Figure 2.2.

reduce the encoding and decoding complexity encountered when utilising the original RLFC, by optimising the sparsity of the coding vector associated with each transmitted packet. This sparsity optimisation meant that an extremely simple decoding algorithm, named message passing or iterative belief propagation (IBP), could be applied.

2.4.1 Iterative Belief Propagation

The IBP decoding process can be said to resemble the simple back substitution algorithm that would be the next phase of the Gaussian Elimination algorithm mentioned in Section 2.2. This is because IBP simply checks the received coding vectors for a degree of one and, if such a vector is found, marks the packet within the vector as recovered. This process then removes (XORs) the newly recovered packet from every other coding vector.

Figure 2.5 shows the IBP algorithm when applied to a toy example such as a scenario where the user has successfully recovered every coded packet generated in Figure 2.2. Packets s_1, s_2 and s_3 are assigned the values of $1, 0, 1$ respectively.

The algorithm initially checks whether any of the coding vectors are systematic, in other words if they have a degree of one. It finds that r_1 has a degree of one and immediately marks s_1 as recovered. The fact that the value of s_1 is 1 is now known, so s_1 can be removed (XORed) from every coded packet which still includes s_1 (packets r_2, r_3, r_4).

When the removal of any s_1 elements in the remaining coding vectors is complete, the IBP algorithm checks again for coded packets of degree one. This time it finds that r_2 is now systematic so, in a similar fashion to the previous step, s_2 is now known to have a value of 0 which is then XORed with the other coding vectors. Two coded packets are then left r_3, r_4 that both claim to contain the value of s_3, and luckily they agree so the algorithm is complete.

However, if the user had *not* received the 4 coded packets that were generated in Figure 2.2, for instance it had not been able to recover r_1, the IBP algorithm would have not been able to recover *any* of the source packets, as there would not have been any degree one packets, even though it is evident from Figure 2.3 that all of the source packets *are* in fact recoverable. It is this limitation of the low computational complexity IBP decoder that the sparsity optimisation schemes described in the next section attempt to mitigate.

2.4.2 Sparsity Optimisation

As mentioned in the previous section, these sparsity optimisation schemes, known as degree distributions, effectively limit the amount of source packets that can be included in each coded packet at each time step. The method adopted in the literature is to employ a probabilistic distribution, where the distribution is designed to meet certain constraints. Formally, the degree distribution must:

- Include every source packet at least once in a coded packet, which means that some packets need to have a large degree.
- Produce enough low-degree packets to keep the IBP process running.
- Keep the average degree low, to minimise the number of encoder and decoder operations needed.
- Keep the average number of coded packets needed to recover the entire source message as low as possible.

Ideal Soliton Degree Distribution

A straightforward way of designing a degree distribution that meets the above constraints would be to engineer a degree distribution where it is expected that there will only be one degree-one packet during every iteration of the algorithm. This distribution is called the Ideal Soliton Distribution [8]:

$$\begin{aligned} \rho(1) &= 1/K \\ \rho(d) &= \frac{1}{d(d-1)} \text{ for } d = 2, 3, \cdots, K \end{aligned} \tag{2.15}$$

where K is the number of source packets.

In practice, the Ideal Soliton Distribution performs poorly. This is because, as the decoding algorithm is waiting for the release of a *single* degree-one coded packet at *every* iteration of the algorithm, any variance in the expected packet degree will halt the recovery process. And, as the degree selection is probabilistic, deviation occurs frequently. It is likely that there will be no degree-one coded packets when expected and it is also likely that some of the K source packets will not be included in any coded packets.

This reliance on the release of a degree-one coded packet at *every* iteration was fixed with the introduction of the Robust Soliton Distribution.

Robust Soliton Degree Distribution

The Robust Soliton Distribution builds on the framework introduced by the Ideal Soliton Distribution by adding two extra parameters, δ and c. Where δ is the bound on the probability that the decoding will eventually fail, known as the failure probability, and c is a constant.

The two new parameters are then optimised to ensure that the expected number of degree-one coded packets at every iteration of the decoding algorithm is not 1, but

$$S \equiv c \log_e(K/\delta)\sqrt{K}. \tag{2.16}$$

Function τ is then defined, in [14], as

$$\tau(d) = \begin{cases} \frac{S}{K}\frac{1}{d} & \text{for } d = 1, 2, \cdots, \lfloor K/S \rfloor - 1 \\ \frac{S}{K}\log_e(S/\delta) & \text{for } d = \lfloor K/S \rfloor \\ 0 & \text{for } d > \lfloor K/S \rfloor. \end{cases} \tag{2.17}$$

The Robust Soliton Distribution is obtained by adding ρ (2.15) to the newly defined τ (2.17) and normalising the result with $Z = \sum_{i=1}^{K}(\rho(i) + \tau(i))$. Hence, the Robust Soliton Distribution (μ) is given as

$$\mu(d) = \frac{\rho(d) + \tau(d)}{Z}. \tag{2.18}$$

In conclusion, the addition of the τ component to the original soliton distribution increases the expected number of degree-one coded packets at each iteration from the impractical value of 1 to S (2.16). Function τ also introduces a spike of coded packets with a larger degree of $\lfloor K/S \rfloor$ so that there is a much more robust chance of every source packet being included in at least one coded packet.

2.5 Summary

In this chapter, the theoretical performance of the Random Linear Fountain Code was examined, with and without the possibility of transmitting a null coded packet. The encoding and decoding process for Luby Transform codes was also looked at.

The next chapter will focus on the introduction of each of the encoding schemes, methods of governing which source packets a RLFC can encode over, and the extension of the theoretical derivation provided in this Chapter to provide exact expressions for some of these encoding schemes. A new metric will be introduced to allow the encoding schemes to be contrasted based on how well they allow receivers to progressively recover source packets, in situations when they have received less than K coded packets.

Chapter 3

Introduction to the Encoding Schemes

In this chapter, each of the encoding schemes that were compared will be introduced. Encoding schemes are methods of governing which source packets a RLFC encoder can combine when generating coded packets. More specifically, the main benefits of systematic fountain codes when compared with non-systematic fountain codes will be looked at.

Utilising a similar rationale to that adopted in Chapter 2, expressions for the theoretical overall performance of both systematic fountain codes and the benchmark scheme, the transmission of ordered uncoded source packets, will be derived.

A new metric will then be introduced, $\mathrm{P}_{K,M}(N)$, that allows the encoding schemes to be contrasted based on how well they allow receivers to progressively recover source packets when they have received less than K coded packets. Where $1 \leq M \leq K$ is the length of the portion of the source message that should be recovered before K coded packets have been received. And finally, expressions for $\mathrm{P}_{K,M}(N)$ for both systematic fountain codes and the benchmark scheme will be developed.

3.1 Conventional Fountain Codes

The first encoding scheme that was utilised was the Random Linear Fountain Code or Conventional Fountain Code (CFC) that has already been touched on in Chapter 2. To reiterate, at every time-step, the CFC encoder produces a coded packet, over the entire source message, that is the bitwise sum (XOR) of a uniformly random number of source packets (2.1).

3.2 Systematic Fountain Codes

If the maximum packet erasure probability of the lossy channel between the transmitter and receiver is relatively low, using a systematic fountain code (SFC) instead of a conventional fountain code can provide a myriad of benefits.

Benefits such as a greatly reduced decoding complexity and a smaller amount of coded packets that need to be received. Also, if enough source packets are received in chronological order, the receiver may be able to start utilising the recovered data, e.g. start playing a multimedia message, before the obligatory K coded packets have been received.

More formally, the SFC that is considered here sequentially transmits each of the K source packets systematically. As soon as every source packet has been transmitted once, the scheme behaves like a CFC. Borrowing from (2.1), the SFC encoder produces a stream of systematic/coded packets where the n-th transmitted packet can be defined as follows

$$t_n = \begin{cases} s_n & \text{if } n \leq K \\ \sum_{i=1}^{K} g_{n,i}\, s_i & \text{otherwise.} \end{cases} \tag{3.1}$$

The theoretical performance of systematic fountain codes will now be investigated by building on the analysis presented in Section 2.3, initially including the possibility of a null-coded packet and subsequently, more practically, removing the possibility of a null-coded packet.

A metric for contrasting the ability of each encoding scheme to progressively recover source packets, in other words the ability to recover and use source packets before K coded packets have been received, will then be introduced. A theoretical expression for the new metric, in the case of SFC, is then introduced.

3.2.1 Theoretical probability of recovering every source packet

For $N = K$, in order to recover each of the K source packets, every transmitted packet must be received. As each of the transmitted packets are systematic, the probability of each received packet being linearly independent does not need to be considered. In a similar manner to the process that produced (2.13), this probability is found to be $\mathrm{P}_K(N = K) = (1 - p)^K$.

When N is larger than K there is a possibility of fountain coded packets being received. To take this into account, the probability that the i received packets are linearly independent now needs to be computed. This is accomplished with the weighted sum of probabilities described in the next section.

Sum of the weighted probability of linear independence of each case

Below is the $(i+1, 2)$ matrix of all the two integer permutations which sum to i for the scenario where $i = 3, K = 3$ and $N = 4$

$$\begin{array}{cc} h & c \\ \end{array}$$
$$\begin{bmatrix} 0 & 3 \\ 1 & 2 \\ 2 & 1 \\ 3 & 0 \end{bmatrix}.$$

The values in the left column signify the number of systematic packets (h) within the i received. The right column signifies the number of fountain coded packets (c).

As the maximum number of systematic packets is K and the maximum number of coded packets is $(N-K)$, some of the configurations above are obviously not valid. In the example, $N = 4$ and $K = 3$ so $h \leq 3$ and $c \leq 1$.

If the rows in the above matrix, that either contain a systematic packet value larger than K or contain a coded packet value larger than $N-K$, are removed,

$$\begin{array}{cc} h & c \\ \end{array}$$
$$\begin{bmatrix} 2 & 1 \\ 3 & 0 \end{bmatrix}$$

is obtained.

The rows of the above matrix now contain valid cases. For example, the first row indicates that there is a possibility of receiving $h = 2$ systematic packets and $c = 1$ coded packet.

Let $\text{Pr}^{S}_{\text{LIN}}(K, i, h)$ denote the probability of the linear independence of the i received packets, of which h are guaranteed to be systematic, which have been encoded over K source packets.

Before summing the individual values of $\text{Pr}^{S}_{\text{LIN}}(K, i, h)$ for each case, each case's probability needs to be weighted by the number of combinations which fit each case. To give an example for the previous scenario, where $i = 3, K = 3$ and $N = 4$, there are $\binom{4}{3} = 4$ possible combinations:

$$\underbrace{\begin{bmatrix} 0 & 1 & 1 & \overbrace{1}^{N-K \text{ cod. packets}} \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}}_{K \text{ sys. packets}}$$

It is evident that only 1 of the combinations fits the $\begin{bmatrix} 3 & 0 \end{bmatrix}$ case, whereas 3 of the combinations fit the $\begin{bmatrix} 2 & 1 \end{bmatrix}$ case. The weighted sum of probabilities for this scenario can now be written as

$$(1 \cdot \Pr^S_{\text{LIN}}(3,3,3)) + (3 \cdot \Pr^S_{\text{LIN}}(3,3,2)).$$

A generalised expression for the number of combinations that fit a certain case is

$$\binom{K}{h}\binom{N-K}{c}.$$

If this expression is summed over every possible value of $h, 0 \leq h \leq K$, and take into account that c is simply $i - h$ it can be confirmed by Vandermonde's convolution [19] that

$$\sum_{h=0}^{K} \binom{K}{h}\binom{N-K}{i-h} = \binom{N}{i}.$$

By introducing the probability of the linear independence of each case, a generalised expression can now be defined for the weighted sum:

$$\sum_{h=0}^{K} \binom{K}{h}\binom{N-K}{i-h} \Pr^S_{\text{LIN}}(K,i,h)$$

Of course, with the above h limits, some of the cases produced are not valid. As an example, there are scenarios where $(i-h) > (N-K)$ such as when $K = 3, N = 4$ and $i = 3$. In other words, 3 systematic packets have been transmitted and only $N - K = 1$ coded packet so, to make up the number of received packets to $i = 3$, $h \geq 2$. To account for this, $h_{min} = \max(0, i - N + K)$.

$$\Pr^S_{\text{LIN}}(K,i) = \sum_{h=h_{min}}^{K} \binom{K}{h}\binom{N-K}{i-h} \Pr^S_{\text{LIN}}(K,i,h) \tag{3.2}$$

	Packet 1	**Packet 2**	**Packet 3**	**Packet 4**
Combination 1	$\frac{0}{8}$	$\frac{8}{8}$	$\frac{6}{8}$	$\frac{4}{8}$
2	$\frac{8}{8}$	$\frac{2}{8}$	$\frac{6}{8}$	$\frac{4}{8}$
3	$\frac{8}{8}$	$\frac{6}{8}$	$\frac{4}{8}$	$\frac{4}{8}$
4	$\frac{8}{8}$	$\frac{6}{8}$	$\frac{4}{8}$	$\frac{8}{8}$

Table 3.1 Possible combinations of $i = 4$ received packets that allows the recovery of the $K = 3$ source packets. The first received packet is guaranteed to be systematic ($h = 1$).

3.2.2 Probability of linear independence of each case

Possibility of a Null coded Packet Included

As ($h \geq 0$) packets are systematic, and hence already linear independent, certain terms in the expression for a CFC in Section 2.3 (2.6) can be omitted.

For example if, in the $i = 4, K = 3$ scenario, the *first* packet was guaranteed to be linearly independent ($h = 1$) the probability of the first packet's linear independence can be set to 1. The four combinations would then become Table 3.1. Again, the darker shaded cells indicate the probability of a linearly *dependent* packet.

Following this train of thought to its conclusion allows the alteration of (2.6) to take into account the possibility of h systematic packets:

$$\Pr_{\text{LIN}}^{S}(K,i,h) = \prod_{a=h}^{K-1}\left(1 - \frac{1}{2^{K-a}}\right) \cdot \sum_{d_1=h}^{K} \frac{2^{d_1}}{2^K} \sum_{d_2=r_1}^{K} \frac{2^{d_2}}{2^K} \quad \cdots \sum_{d_{(i-K)}=d_{(i-K)-1}}^{K} \frac{2^{d_{(i-K)}}}{2^K}. \tag{3.3}$$

Possibility of a Null coded Packet Removed

Following the same process that produced (3.3), but starting with (2.7) instead of (2.2) $\Pr_{\text{LIN}}^{S}$ can be redefined as:

$$\Pr_{\text{LIN}}^{S}(K,i,h) = \prod_{a=h}^{K-1}\left(1 - \frac{2^a - 1}{2^K - 1}\right) \cdot \\ \cdot \sum_{d_1=h}^{K} \frac{2^{d_1}-1}{2^K-1} \sum_{d_2=d_1}^{K} \frac{2^{d_2}-1}{2^K-1} \quad \cdots \sum_{d_{(i-K)}=d_{(i-K)-1}}^{K} \frac{2^{d_{(i-K)}}-1}{2^K-1}. \tag{3.4}$$

where, as there is no possibility of a null coded packet, $h \geq 1$.

By taking $\frac{1}{(2^K-1)^{i-K}}$ as a common factor, (3.4) can be simplified to:

$$\begin{aligned}\mathrm{Pr}^{S}_{\mathrm{LIN}}(K,i,h) =& \frac{1}{(2^K-1)^{i-K}} \prod_{a=h}^{K-1} \left(1 - \frac{2^a-1}{2^K-1}\right) \cdot \\ & \cdot \sum_{d_1=h}^{K} 2^{d_1} - 1 \sum_{d_2=d_1}^{K} 2^{d_2} - 1 \ \cdots \sum_{d_{(i-K)}=d_{(i-K)-1}}^{K} 2^{d_{(i-K)}} - 1. \quad (3.5)\end{aligned}$$

If $\mathrm{Pr}^{S}_{\mathrm{LIN}}(K,i,h)$ is substituted into (3.2), the complete expression for the weighted sum of the probability of linear independence of each case is found:

$$\begin{aligned}\mathrm{Pr}^{S}_{\mathrm{LIN}}(K,i) = & \sum_{h=h_{min}}^{K} \binom{K}{h}\binom{N-K}{i-h} \cdot \frac{1}{(2^K-1)^{i-K}} \prod_{a=h}^{K-1} \left(1 - \frac{2^a-1}{2^K-1}\right) \cdot \\ & \cdot \sum_{d_1=h}^{K} 2^{d_1} - 1 \sum_{d_2=d_1}^{K} 2^{d_2} - 1 \ \cdots \sum_{d_{(i-K)}=d_{(i-K)-1}}^{K} 2^{d_{(i-K)}} - 1. \quad (3.6)\end{aligned}$$

If the probability of actually receiving i packets is then introduced, over every possible value of i, into (3.6) the full expression for the probability of recovering K source packets after transmitting N coded packets over a packet erasure channel with erasure probability p is:

$$\begin{aligned}\mathrm{P}_K(N) &= \sum_{i=K}^{N} (1-p)^i p^{N-i}\, \mathrm{Pr}^{S}_{\mathrm{LIN}}(K,i) \\ &= \sum_{i=K}^{N} (1-p)^i p^{N-i} \sum_{h=h_{min}}^{K} \binom{K}{h}\binom{N-K}{i-h} \cdot \frac{1}{(2^K-1)^{i-K}} \prod_{a=h}^{K-1} \left(1 - \frac{2^a-1}{2^K-1}\right) \cdot \\ & \cdot \sum_{d_1=h}^{K} 2^{d_1} - 1 \sum_{d_2=d_1}^{K} 2^{d_2} - 1 \ \cdots \sum_{d_{(i-K)}=d_{(i-K)-1}}^{K} 2^{d_{(i-K)}} - 1. \quad (3.7)\end{aligned}$$

3.2.3 Probability of recovering at least M packets

Up to this point the investigation has been focussing on scenarios where $N \geq K$ where the receiver has been attempting to recover *every* source packet.

From this point onwards, *any* value of N and the possibility of recovering at least M source packets out of K will be considered. This is accomplished by introducing a metric that allows the comparison of the ability of each encoding scheme to recover and possibly utilise an amount of source packets larger or equal to M, before K coded packets have been received.

Probability $\mathrm{P}_{K,M}(N)$ is defined as the probability that at least M source packets from subset $\{s_1, \ldots, s_m\}$ have been recovered, given that packets $t_1, t_2, \ldots, t_N$ have been transmitted, where $M \leq m \leq \min(K, N)$. For example, if a message is comprised of source packets $s_1, \ldots, s_{10}$ and the encoder transmits packets $t_1, \ldots, t_6$

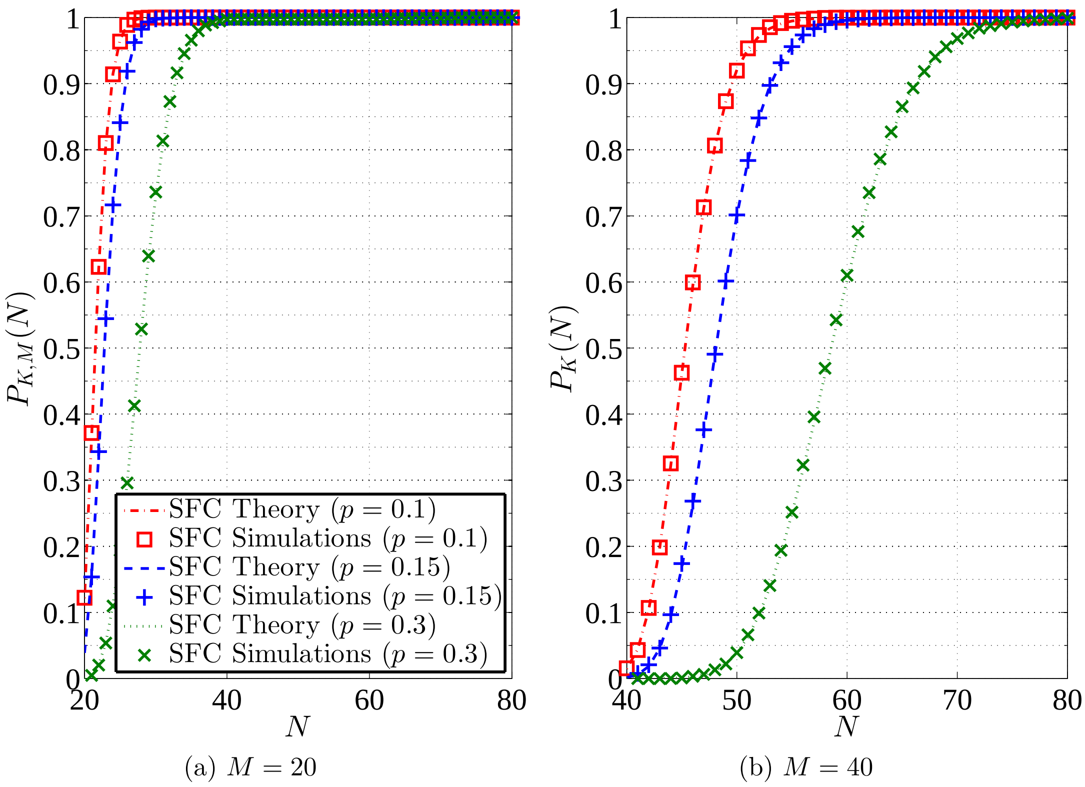


Fig. 3.1 Performance validation of SFC transmission for $K = 40$, different values of p and (a) partial message recovery ($M = 20$) or (b) full message recovery ($M = 40$).

in six time steps. Probability $\mathrm{P}_{K=10,M}(N = 6)$ would focus on the recovery of M or more source packets from subset $\{s_1, \ldots, s_6\}$, even if source packets that come after s_6 in chronological order have been recovered at the destination. From this point on, $\mathrm{P}_{K,M}(N)$ is referred to as the "progressive" performance of an encoding scheme.

The probability of recovering at least M packets in the range of $\{s_1, \ldots, s_N\}$ is simplified with the adoption of SFC, as the first K transmitted packets are all systematic.

$$\mathrm{P}_{K,M}(N) \approx \sum_{i=M}^{N_{min}} \binom{N_{min}}{i} (1-p)^i p^{N_{min}-i} \tag{3.8}$$

where $N_{min} = \min(K, N)$.

This expression is always exact for cases where $M < (1 - p) \cdot K$ and exact for other cases up to $N = K$, where it becomes a lower bound.

Figure 3.1 compares analytical results with simulation results for the SFC scheme with differing channel erasure probabilities. It is observed that the theoretical expressions presented, accurately match the simulated results for both progressive and overall packet recovery.

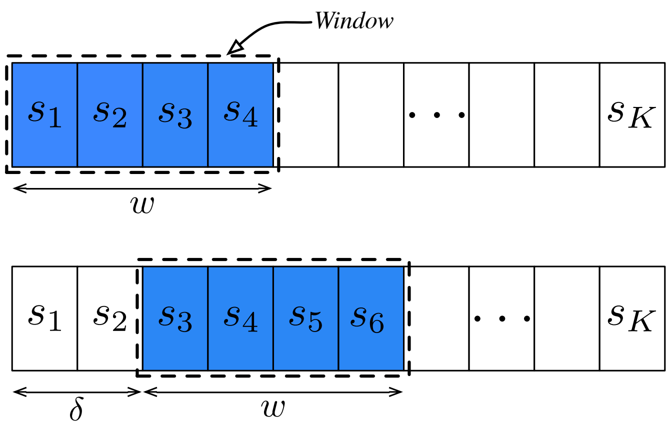

Window
s_1 s_2 s_3 s_4 ... s_K
w
s_1 s_2 s_3 s_4 s_5 s_6 ... s_K
δ
w

Fig. 3.2 SWFC scheme as proposed in [2], [3]. In this case, $w = 4$ and $\delta = 2$, this means that every window is encoded over for $w = 4$ transmissions [1].

3.3 Sliding-Window Fountain Codes

The Sliding-Window fountain code (SWFC) scheme was initially presented by Bogino *et al.* in [2] and then expanded in [3]. It attempts to address the intrinsic weaknesses of a CFC when employed to stream multimedia content, in that packets are not necessarily received in chronological order, by introducing a windowed element.

This new window is governed by two additional system parameters; the size of the window w, in terms of source packets, and the number of packets δ that the window traverses when it slides after w transmissions. When the window has been finalised, a CFC is applied over the selected portion of the source message only.

For example, in the scenario presented in Figure 3.2, where $w = 4$ and $\delta = 2$, the first $w = 4$ transmitted packets $\{t_n\}_{n=1}^{4}$ will be encoded over s_1, s_2, s_3 and s_4, that is $t_n = g_{n,1}\, s_1 + g_{n,2}\, s_2 + g_{n,3}\, s_3 + g_{n,4}\, s_4$ for $n = 1, \ldots, 4$. Coefficients $g_{n,i}$ are binary numbers selected from set $\{0, 1\}$ uniformly at random, as mentioned in Section 2.1. The next $w = 4$ transmitted packets $\{t_n\}_{n=5}^{8}$ will be encoded over s_3, s_4, s_5 and s_6 in a similar fashion.

More formally, let s_ℓ and s_r be the leftmost and rightmost source packets included in the window, the coded packet t_n, produced at time step n, can then be defined as

$$t_n = \sum_{i=\ell}^{r} g_{n,i}\, s_i\ . \tag{3.9}$$

where

$$\ell = \delta \left\lfloor \frac{n-1}{w} \right\rfloor + 1 \tag{3.10}$$

and $r = \ell + w - 1$.

However, as the original SWFC scheme was engineered for *stream-based* multimedia content delivery, this scheme's behaviour after the *end* of a multimedia stream has not been defined. This is because the scheme's authors assume that a new source message block would be presented before/at the end of the current block's transmission.

This investigation is interested in the performance of each of the encoding schemes over a single source message block, so the SWFC scheme defaults to the CFC described in Chapter 2 after the right edge of the window encompasses the K^{th} source packet. In other words, for $r = K$ the SWFC scheme becomes (2.1). This is to maximise the probability of recovery of the source packets that were not successfully received in the Sliding-Window phase.

3.3.1 Optimising the window's parameters

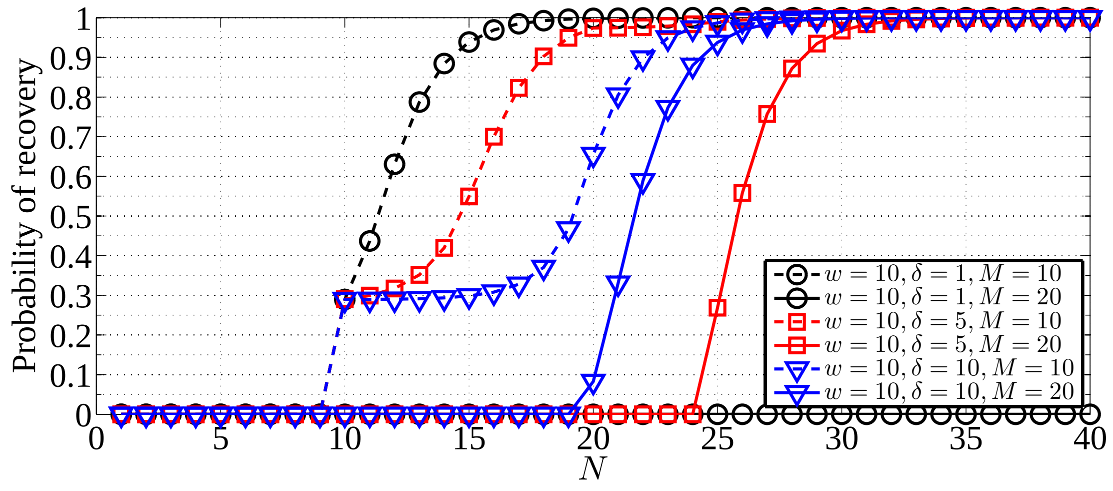
In the implementation of the SWFC scheme, in order to maximise the progressive performance, $w = M$ and $\delta = M/2$ as this configuration allows the scheme to initially resemble a "miniature" CFC over the first M source packets. This provides enhanced progressive performance while minimising the delay in recovering every source packet, a trade-off that will be explored next.

Fixed window size w

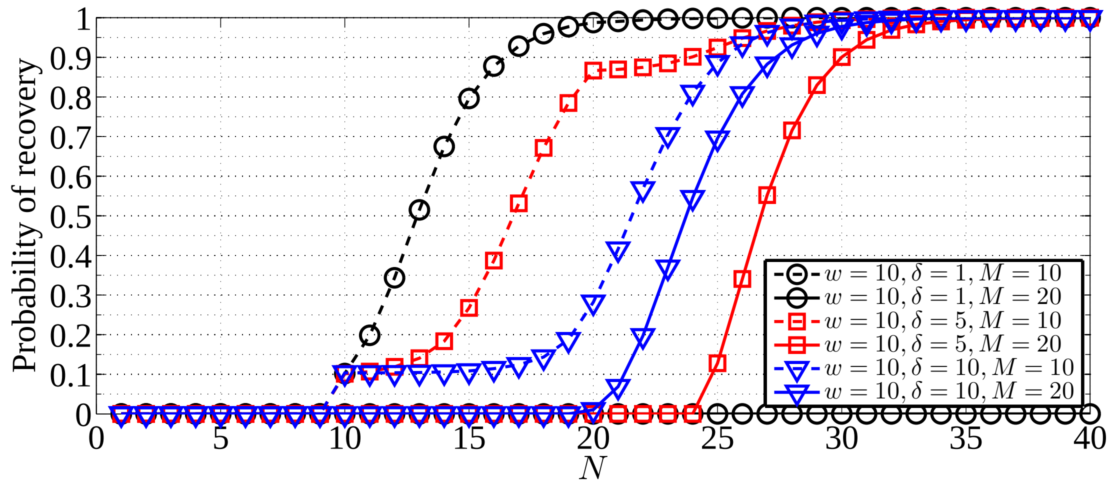
Figure 3.3 shows, for a constant window size, the effect that altering the value of δ has on both the progressive and full recovery of the $K = 20$ source packets. Although every scheme's progressive performance is identical at $N = 10$, once the different δ values start to influence each scheme's performance, it is possible to see the trade-off between progressive and overall performance when choosing w and δ values.

It is apparent that a low value of $\delta = 1$ leads to excellent progressive performance as there is a 9 packet overlap between consecutive windows that results in the first 10 source packets being included in numerous coded packets. However, this δ value means that it takes the scheme over 100 coded packets, or 10 window iterations, to traverse the entire source message which results in unacceptable overall performance.

For a larger value of $\delta = 10$ the scheme exhibits reduced progressive performance and increased overall performance when compared to $\delta = 1$. This is because, after the first 10 coded packets have been encoded over the first 10 source packets, the scheme switches straight to a CFC that encodes over the entire source message. Because the progressive metric attempts to find the probability of recovering *at least* $M < K$ source packets *in the range of* $\{s_1, \ldots, s_N\}$, where $N > M$, instead of the probability of recovering at least *any* M, the progressive performance then remains constant.



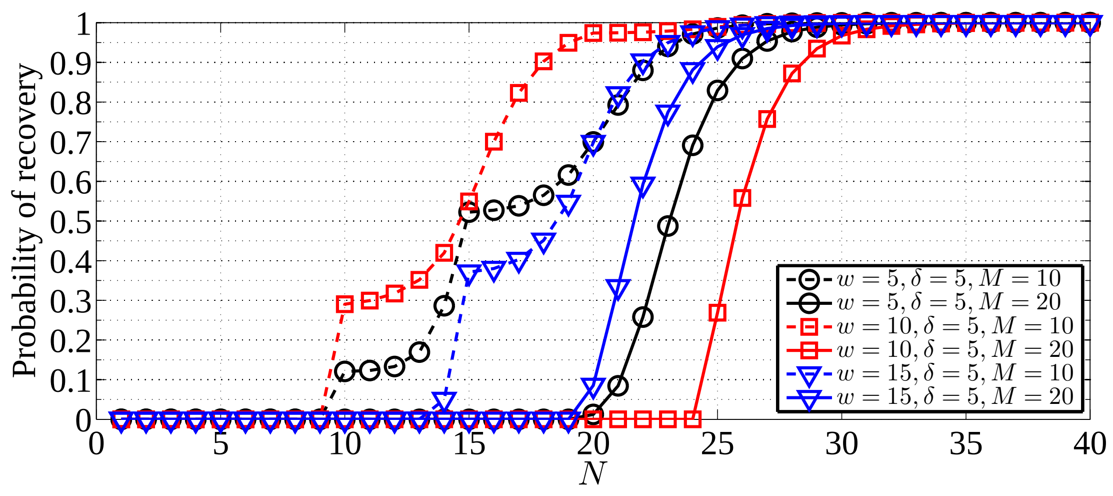
(a) $p = 0.001$



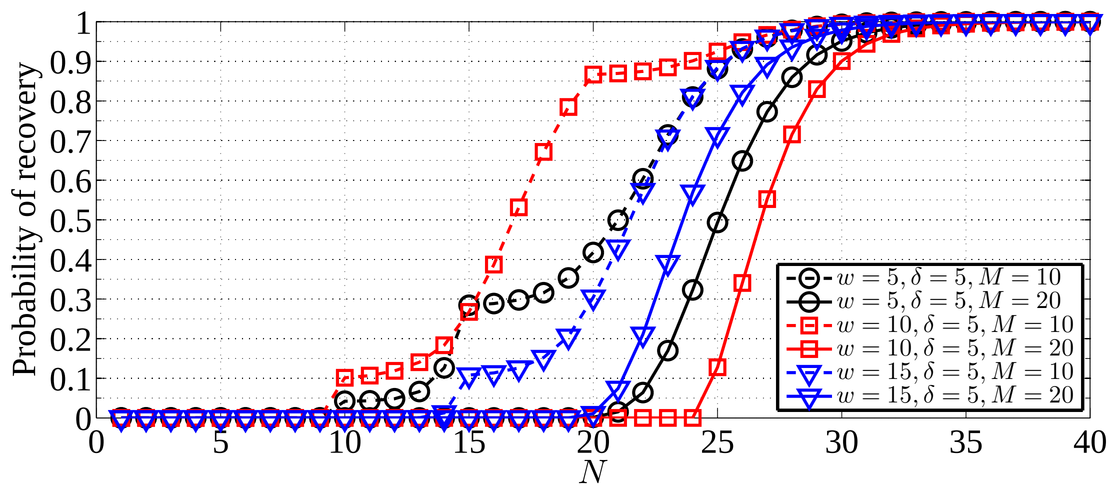
(b) $p = 0.1$

Fig. 3.3 Performance of the SWFC scheme for $K = 20$, different values of δ with a fixed w and (a) 0.1% packet erasure and (b) 10% packet erasure.

As an example, for $N = 11, w = 10$ and $\delta = 10$, the progressive metric is concerned with the probability of recovering *at least* M source packets in the range of $\{s_1, \ldots, s_{11}\}$. The progressive performance does not increase as the probability of successfully recovering s_{11} from a CFC coded packet, encoded over the entire source message, is very low as there are too many unknown elements in the coding vector. This reasoning continues until enough new coded packets, that have been encoded over the entire source message, have been received from $N = 18$ onwards. The switch to a CFC after the 10^{th} transmitted coded packet means that every source packet has a chance of being included in a coded packet sooner, when compared to the $\delta = 1$ scenario. This results in better overall performance.



(a) $p = 0.001$



(b) $p = 0.1$

Fig. 3.4 Performance of the SWFC scheme for $K = 20$, different values of w with a fixed δ and (a) 0.1% packet erasure and (b) 10% packet erasure.

Fixed δ value

In contrast to Figure 3.3, Figure 3.4 shows the different behaviours exhibited by the SWFC scheme for different values of window size, while δ is kept constant.

If the case with a smaller value of $w = 5$ and $\delta = 5$ is examined, at $N = 10$ poorer progressive performance is observed than that of the $w = 10$ and $\delta = 5$ case. This is because the progressive performance at that point depends on receiving *every* coded packet produced by two CFC codes, applied to smaller sets of 5 source packets each, and each set of the coded packets produced by the two smaller codes being linearly independent of each other.

Whereas the $w = 10$ and $\delta = 5$ case also needs to recover every coded packet, but these coded packets have been encoded over a window that is twice the size, ($K = 10$), which increases the probability of the 10 packets being linearly independent.

If the case where $w = 10$ is considered, it is observed that there is a delay in overall performance when compared to the other two configurations. This is a result of the switch to a CFC being after $N = 20$, because the last δ packets have not had a chance of being included in any coded packets at this point. As there are now $\delta = 5$ new variables (source packets) to consider, the receiver needs at least $\delta = 5$ innovative simultaneous equations (coded packets) to even stand a chance of recovering them.

The $w = 15$ and $\delta = 5$ case's performance is the closest in this set to a conventional fountain code's response. The overall performance is better than the other two configurations as it has been encoding over a larger source packet pool for longer, which results in each coded packet having a larger probability of linear independence. However, encoding over this larger packet pool means that it is harder to pick out the first half of the source message, so the progressive performance suffers.

To conclude, the $w = 10, \delta = 5$ was opted for or, in general, the $w = M, \delta = M/2$ configuration, as it offered the best progressive performance whilst minimising delay in overall performance.

3.4 Ordered Uncoded Scheme

The Ordered Uncoded (OU) scheme will be used as a benchmark throughout the investigation. Its main aim is to assess whether there is actually any benefit, from a progressive performance viewpoint, to adding coding complexity to both the encoder and the decoder.

As the name of this strategy implies, this scheme simply transmits each of the K source packets systemically, in chronological order, and then repeats itself. In similarity to the SFC analysis above, theoretical expressions will now be developed for this benchmark scheme for both progressive and overall packet recovery.

3.4.1 Theoretical Probability of Recovering Every Source Packet

If a source message consists of K packets that are transmitted N times under the Ordered Uncoded scheme, N could be written as $N = \alpha K + \beta$ where $\alpha = \lfloor N/K \rfloor$ and $\beta = (N \bmod K)$. In other words the sender has transmitted:

- $(\alpha + 1)$ copies of packets $s_1, \dots, s_\beta$
- α copies of packets $s_{\beta+1}, \dots, s_K$

If the previous chapter's definition of the probability of packet erasure p is utilised, then again the probability of recovering a single packet that has been transmitted *once* is $(1-p)$. If this single packet was transmitted *twice*, there are 2^2-1 possible situations that would result in the recovery of the packet.

- The first transmission is recovered but not the second with a probability of $(1-p)p$.
- The second transmission is recovered but not the first with a probability of $p(1-p)$.
- *Both* transmissions are recovered with a probability of $(1-p)(1-p)$.

As each of these scenarios occur with an equal probability, they are simply summed to obtain the full probability of recovering a single source packet transmitted twice:

$$\begin{aligned}
&= (1-p)p + p(1-p) + (1-p)(1-p) \\
&= p - p^2 - p^2 + p + 1 - p - p + p^2 \\
&= 2p - 2p - p^2 - p^2 + p^2 + 1 \\
&= \cancel{2p} - \cancel{2p} - p^2 - \cancel{p^2} + \cancel{p^2} + 1 \\
&= 1 - p^2
\end{aligned}$$

It follows that the probability of recovering a single packet that has been transmitted three times is $1-p^3$ and n times is $1-p^n$.

The probability of recovering three different packets that have each been transmitted three times, that is $N = 3K$, can now be written as $(1-p^3)^3$. In general, the probability of recovering the K source packets from the N received packets can be written as follows:

For $N = K$, the probability of recovering all the source packets is

$$\mathrm{P}_K(K) = (1-p)^K . \tag{3.11}$$

For $N = K + 1$,

$$\mathrm{P}_K(K+1) = \left(1-p^2\right)(1-p)^{K-1} . \tag{3.12}$$

For $N = K + \beta$ where $0 \le \beta < K$,

$$\mathrm{P}_K(K+\beta) = \left(1-p^2\right)^{\beta}(1-p)^{K-\beta} . \tag{3.13}$$

For $N = \alpha K + \beta$,

$$\mathrm{P}_K(N) = \left(1 - p^{\alpha+1}\right)^{\beta} (1 - p^{\alpha})^{K-\beta}. \tag{3.14}$$

3.4.2 Probability of recovering exactly m source packets

The initial step in finding a theoretical expression for the probability of recovering *at least* M source packets is to find an expression for the probability of recovering *exactly* m source packets. Once this is accomplished, this expression can be extended for a specific instance of m for a more general case where $M \leq m \leq K$.

It follows from (3.14) that, if $0 \leq h \leq m$ is defined as the number of "privileged" source packets that have been transmitted $\alpha + 1$ times, the starting point for the probability of recovering exactly m packets is

$$f(m,h) = \left(1 - p^{\alpha+1}\right)^{h} \left(1 - p^{\alpha}\right)^{m-h}.$$

The probability of *not* recovering the other $K - m$ source packets now needs to be taken into account. Some of these packets have been transmitted $(\alpha + 1)$ times and others just α times.

h "privileged" packets have already been accounted for, so there still remain $\beta - h$ source packets with $\alpha + 1$ copies that need to be addressed.

The $K - \beta - (m - h)$ "unprivileged" source packets with α copies also need to be addressed. The total number of packets that have not been recovered with probability p is given by:

$$\begin{aligned}
&\overbrace{(\beta - h)(\alpha + 1)}^{\text{Privileged}} + \overbrace{(K - \beta - (m - h))(\alpha)}^{\text{Unprivileged}} \\
&= \beta\alpha + \beta - h\alpha - h + K\alpha - \beta\alpha - m\alpha + h\alpha \\
&= \cancel{\beta\alpha} + \beta - \cancel{h\alpha} - h + K\alpha - \cancel{\beta\alpha} - m\alpha + \cancel{h\alpha} \\
&= \beta - h + \alpha(K - m)
\end{aligned}$$

An expression for a *specific instance* of the probability of recovering m source packets can now be written:

$$f(m,h) = \left(1 - p^{\alpha+1}\right)^{h} \left(1 - p^{\alpha}\right)^{m-h} p^{\beta-h+\alpha(K-m)}. \tag{3.15}$$

However, (3.15) is not complete, as the fact that within the first β and the last $K - \beta$ there are a multitude of ways that h and $m - h$ packets can be organised needs to be taken into account.

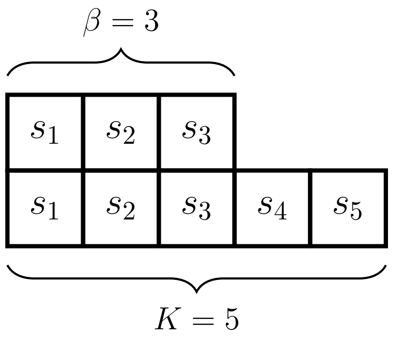


Fig. 3.5 An example showing $K = 5$ source packets, three of which ($\beta = 3$) have been transmitted twice and are considered "privileged". If exactly $m = 3$ packets are received and h_{min} started from 0, this would imply that there are 3 "unprivileged" packets. Hence, h_{min} must start from 1.

In fact, there are

$$\binom{\beta}{h}$$

ways of selecting h different privileged packets from β possible choices.

And there are

$$\binom{K-\beta}{m-h}$$

ways of selecting $m-h$ different unprivileged packets from $K-\beta$ possible choices.

In conclusion, for a set of valid m and h values there are

$$\binom{\beta}{h}\binom{K-\beta}{m-h}$$

possible combinations.

Hence, the probability of recovering *exactly* m source packets, of which h packets are privileged, is

$$\mathrm{P}_{K,m,h}(N) = \binom{\beta}{h}\binom{K-\beta}{m-h}\left(1-p^{\alpha+1}\right)^{h}\left(1-p^{\alpha}\right)^{m-h} p^{\beta-h+\alpha(K-m)}. \tag{3.16}$$

To find the complete probability of recovering exactly m source packets then, as there are multiple valid values for h, $\mathrm{P}_{K,m,h}(N)$ must be summed over every valid value of h:

$$\mathrm{P}_{K,m}(N) = \sum_{h=h_{min}}^{h_{max}} \binom{\beta}{h}\binom{K-\beta}{m-h}\left(1-p^{\alpha+1}\right)^{h}\left(1-p^{\alpha}\right)^{m-h} p^{\beta-h+\alpha(K-m)}. \tag{3.17}$$

Obviously, $h_{max} = \min(\beta, m)$. This is because h, the number of privileged packets, cannot be larger than the actual number of privileged packets or the number of packets this iteration is interested in.

On the other hand, $h_{min} = \max(0, m - K + \beta)$. For example, after examining Figure 3.5 with $K = 5$ and $\beta = 3$, and assuming that $N = K + \beta = 8$ packets have been transmitted and $m = 3$ packets have been received, it is apparent that the number of received privileged packets h *must* start from 1 as, in this specific scenario, if it started from 0 that would imply that there are $m = 3$ unprivileged received packets. This would be an incorrect assumption as only $K - \beta = 2$ unprivileged packets actually exist.

3.4.3 Probability of Recovering At Least *M* Packets

During the previous section, an expression for the probability of recovering exactly m packets was introduced. In order to obtain the probability of recovering at least M packets, $\mathrm{P}_{K,m}(N)$ must be summed over all the valid values of m. e.g. $M \leq m \leq K$

$$\mathrm{P}_{K,M}(N) = \sum_{m=M}^{K} \mathrm{P}_{K,m}(N). \tag{3.18}$$

By substituting (3.17) into (3.18):

$$\mathrm{P}_{K,M}(N) = \sum_{m=M}^{K} \sum_{h=h_{min}}^{h_{max}} \binom{\beta}{h} \binom{K-\beta}{m-h} \Big(1-p^{\alpha+1}\Big)^{h} \Big(1-p^{\alpha}\Big)^{m-h} p^{\beta-h+\alpha(K-m)} \tag{3.19}$$

Figure 3.6 compares analytical results with simulation results for the OU scheme with differing channel erasure probabilities. It is observed that the theoretical expressions presented accurately match the simulated results for both progressive and overall packet recovery. These theoretical expressions will be utilised as a benchmark in later chapters to assess the performance of the three encoding schemes under consideration, that is CFC, SFC and SWFC.

3.5 Summary

In this chapter, each of the encoding schemes were introduced and theoretical expressions for the overall performance of certain schemes were developed. A metric that was used to assess the progressive capability of each scheme was introduced, and again theoretical expressions for certain schemes were provided.

The next chapter is concerned with the decoding algorithm that was adopted to provide quality simulation results in a timely manner. Fountain code decoding algorithms that were already present in literature are compared, and the enhancements that made a particular scheme, known as Adapted Gaussian Elimination, the most suitable are discussed.

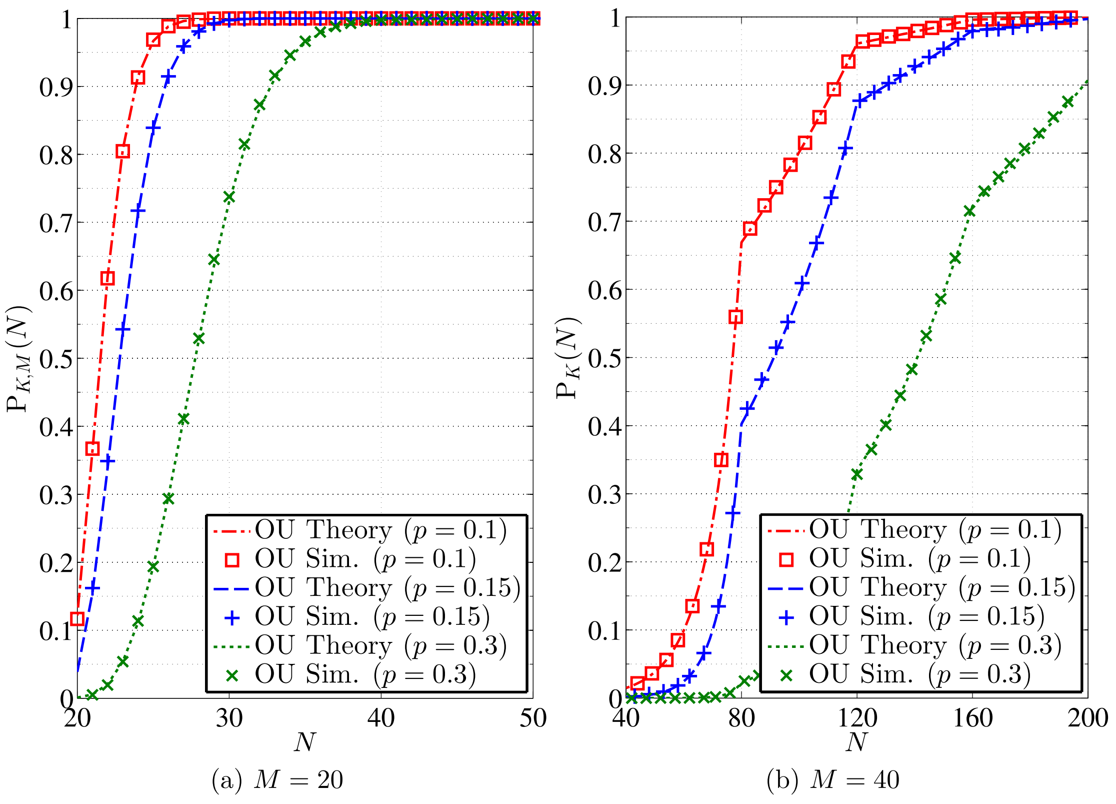


Fig. 3.6 Performance validation of OU transmission for $K = 40$, different values of p and (a) partial message recovery ($M = 20$) or (b) full message recovery ($M = 40$).

Chapter 4

Decoding Schemes for Progressive Recovery

In this chapter, the performance of both Iterative Belief Propagation and the Gaussian Elimination algorithm mentioned in Chapter 2 are compared, whilst examining the suitability of a new decoding strategy called On-the-fly Gaussian Elimination.

4.1 On-the-fly Gaussian Elimination

In addition to the Gaussian Elimination (GE) and Iterative Belief Propagation (IBP) decoding techniques described in Chapter 2, numerous other fountain code decoding algorithms have been proposed.

Iterative Gaussian Elimination (IGE) [20] attempts to reduce the computational expense of GE by only performing a costly "full" triangulation step *once*, at the point when K coded packets have been received. Each subsequent received packet is then incorporated into a *partially* decoded coding matrix.

Concatenated schemes such as Joint Belief Propagation and Gaussian Elimination (JBPGE) [21] try to trade-off the decoding speed of IBP against the reduction in the number of coded packets needed to successfully decode a source message provided by GE. This, of course, does lead to an increase in algorithmic complexity.

On-the-fly Gaussian Elimination (OFGE) [4] continues to mitigate the decoding delay and computation complexity of GE by invoking an optimised triangulation process *every time* a coded packet is received.

The OFGE algorithm was chosen for inclusion in the simulation model as it offered multiple advantages over its competitors. For example, when compared to IGE, it is evident that the OFGE decoder spreads its computation out over each packet arrival, while IGE has a large spike in computation when K packets are received. This more efficient “spacing out” of computation also means that when K packets have been received, OFGE already has a partially triangular decoding matrix, this reduces the delay in recovering each of the source packets as fewer row operations are required.

The main reason for this choice of decoding algorithm over IBP, apart from the reduction in overhead for the relatively small K values that this investigation is considering, was the fact that IBP usually relies on a degree distribution to function. The decision to examine the worst case performance of fountain codes by using a *random* degree distribution meant that it would be very unlikely, as K grows, that there would be enough degree-one coded packets for IBP to perform.

The original OFGE algorithm presented in [4] will now be described, and subsequently the adapted version that was actually utilised to obtain the simulation results presented in [1]. In the late stages of the project, in order to obtain *exact* simulation results, the suitability of the original GE process mentioned in Chapter 2 was reconsidered. The modifications made to ensure the *exact* GE process was as efficient as possible are then discussed.

4.1.1 Original Algorithm

Listing 4.1, provides a MATLAB implementation of the original OFGE algorithm proposed in [4]. In the implementation, the states of `G` and `degree_of_G` are preserved over each iteration of the algorithm, in other words, between each coded packet arrival.

The algorithm is now worked through with a toy example, during this process the limitations of the original algorithm when used to decode rank-deficient coding matrices will be encountered. These limitations are to be expected, as the original algorithm was not engineered to produce an exact result *before* K coded packets had been received. Instead, it was produced to reduce the delay in decoding a *full-rank* decoding matrix by spreading the computation over each packet arrival, as opposed to waiting until K coded packets have been received before attempting to decode the coding matrix from scratch.

An example of how the original OFGE algorithm performs when provided with rank-deficient coding vectors is now presented, assuming that the encoder is using a Random Linear Fountain Code, as described in Chapter 2, to generate the coded packets. The source is comprised of $K = 5$ source packets. It is plausible that the first two coded packets that the receiver encounters are characterised by the coding vectors $\mathbf{g}_1 = [1, 1, 0, 0, 1]$ and $\mathbf{g}_2 = [1, 1, 1, 1, 0]$. The decoding variable `G` initially denotes a K by K matrix of zeros, and `G`$[t]$ signifies the t-th row of `G`.

When $\mathbf{g}_1$ is received, it is immediately placed into the first row of `G` (`G`[1]) as

```matlab
% New packet arrives
NewEq = new_encoded_packet;

% What is the position of the leftmost 1 in the new packet?
s = find(NewEq,1);

% What is the degree of the new packet?
EqOnes = nnz(NewEq);

% While NewEq has not been xored into nothing and there is
% already a coded packet in row s
while ( EqOnes > 0 ) && (G(s,s) == 1)

    % if degree of new packet is same or larger than row s
    if EqOnes >= degree_of_G(s)
        % xor with the current occupant of row s
        NewEq = xor(NewEq, G(s,:));
    else
        % if the new packet's degree is less than the packet
        % currently in row s then swap them and start
        % comparing with the old row (currently G(s,:))
        temp = G(s,:);
        G(s,:) = NewEq;
        NewEq = temp;
        degree_of_G(s) = EqOnes;
    end

    % and refresh s and EqOnes to continue comparing with
    % other rows, and move on to the next iteration
    s = find(NewEq,1);
    EqOnes = nnz(NewEq);
end

% this is triggered when there are currently no encoded
% packets in row s - the newEq is placed straight into
% G matrix at the correct row
if(EqOnes > 0)
    G(s,:) = NewEq;
end
```

Listing 4.1 MATLAB example of the original OFGE algorithm presented in [4].

it is the first coding vector to be received that has s_1 as its leftmost non-zero element. With the introduction of $\mathbf{g}_2$ a slightly more complex operation must be carried out. This is because $\mathbf{g}_1$ and $\mathbf{g}_2$ share s_1 as their leftmost non-zero element. As $\mathtt{G}$ is being transformed into a row-echelon form matrix, two coding vectors with identical leftmost non-zero elements cannot exist in $\mathtt{G}$.

This situation is mitigated by either XORing the two competing vectors together and then attempting to place the resultant vector elsewhere in $\mathtt{G}$, or by swapping the two competing vectors and then XORing them together. The decision to swap the two competing vectors before the XORing takes place is governed by whether the more recent vector has a lower degree, or total number of non-zero elements, than the competing vector. This addition to the algorithm results in a much sparser matrix by the time that K coded packets have been received, which eventually results in fewer XOR operations being required to recover each of the source packets.

In this case, no swapping takes place as $\mathbf{g}_2$ does *not* have a lesser degree than the vector already present in $\mathtt{G}[1]$. $\mathbf{g}_2$ is replaced by $\mathtt{G}[1] \oplus \mathbf{g}_2 = [1,1,0,0,1] \oplus [1,1,1,1,0] = [0,0,1,1,1]$ and is immediately placed in the third row of $\mathtt{G}$ as the first non-zero element of the redefined vector $\mathbf{g}_2$ is s_3 and $\mathtt{G}[3]$ was previously empty. At this point $\mathtt{G}$ would look like the left hand side of the following relationship

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{Recv.\, \mathbf{g}_3} \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{4.1}$$

If coding vector $\mathbf{g}_3 = [1,0,1,1,1]$ is then received, it will be replaced with $\mathtt{G}[1] \oplus \mathbf{g}_3 = [1,1,0,0,1] \oplus [1,0,1,1,1] = [0,1,1,1,0]$ as it has a larger degree than the vector currently in $\mathtt{G}[1]$. The new value for $\mathbf{g}_3$ is then placed in the second row of $\mathtt{G}$ as $\mathtt{G}[2]$ was previously empty, namely, the right hand side of (4.1).

At this point, $\mathbf{g}_4 = [0,1,1,0,1]$ is recovered. Once again, new vector $\mathbf{g}_4$ does not have a lesser degree than the vector currently occupying $\mathtt{G}[2]$, and is replaced with $\mathtt{G}[2] \oplus \mathbf{g}_4 = [0,1,1,1,0] \oplus [0,1,1,0,1] = [0,0,0,1,1]$. This new value for $\mathbf{g}_4$ is placed directly into $\mathtt{G}[4]$. $\mathtt{G}$ would now look like the left hand side of the following relationship

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{Recv.\, \mathbf{g}_5} \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.2}$$

Finally, $\mathbf{g}_5 = [1,1,1,1,1]$ is recovered. $\mathbf{g}_5$ has a larger degree than the vector currently in $\mathtt{G}[1]$, so it is replaced by $\mathtt{G}[1] \oplus \mathbf{g}_5 = [1,1,0,0,1] \oplus [1,1,1,1,1] = [0,0,1,1,0]$. The new value of $\mathbf{g}_5$ is now compared against the vector currently in $\mathtt{G}[3]$, and is found to have a lesser degree. The vectors $\mathbf{g}_5$ and $\mathtt{G}[3]$ are now swapped in order to increase the sparsity of G.

The new value of $\mathbf{g}_5 = [0,0,1,1,1]$ means that it has a larger degree than that of the vector in $\mathtt{G}[3]$ so it is replaced by $\mathtt{G}[3] \oplus \mathbf{g}_5 = [0,0,1,1,0] \oplus [0,0,1,1,1] = [0,0,0,0,1]$. This new value for $\mathbf{g}_5$ is then placed directly into $\mathtt{G}[5]$. At this point, G would look like the right hand side of (4.2).

Now that G is full-rank it is straightforward to recover the individual source packets via a simple back-substitution stage, such as the process mentioned in Chapter 2. For example, s_5 is known so remove it from the other coding vectors, which leaves s_4 and so on.

However, if the simple back-substitution stage was applied to G at any point other than when 5 coded packets had been received, no source packets would have been recovered, as there were no degree one coding vectors.

In order for the progressive packet recovery metric to function, it requires knowledge of which packets can be recovered *before* K coded packets have been received. It is this requirement that the customised OFGE implementation described in the next section attempts to satisfy.

4.1.2 Adapted Algorithm

As mentioned in the above section, the original OFGE algorithm encounters accuracy issues when utilised to extract which source packets are recoverable from a rank-deficient coding matrix. For example, after $\mathbf{g}_3$ has been received, the right hand side of (4.1), it is possible to recover s_1 by XORing rows $1, 2$ and 3 together. In addition, after $\mathbf{g}_4$ has been received, the left hand side of (4.2), s_3 can also be recovered by XORing rows 3 and 4 together.

In order to uncover which source packets can be recovered from a rank-deficient coding matrix, the original OFGE algorithm was adapted to include an XORing stage that examines the output of the original algorithm after each coded packet arrival. Initially, the algorithm was recursive and exact, in that it checked *every* possible combination of coding vectors, however, this was extremely computationally expensive.

In an attempt to reduce the amount of time that the simulations took to run, a trade-off was made regarding the accuracy of the adapted algorithm and, hence, a limit of 1 was imposed on the recursive depth.

Figure 4.1 shows an example of the combinations that would be checked during

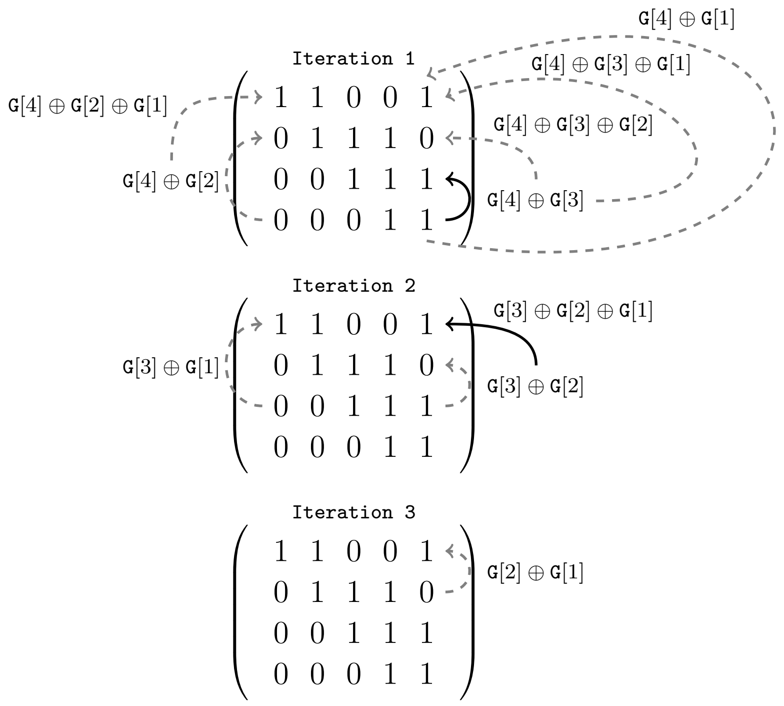


Fig. 4.1 An example showing the combinations that would be checked during each iteration of the OFGE XORing stage after $\mathbf{g}_4$ has been received.

each iteration of the XORing stage after $\mathbf{g}_4$ has been received. The algorithm starts from the furthest row down and ignores empty rows. The light grey dashed combinations do *not* result in a source packet recovery. The recursion depth limit is evident when examining the $\mathtt{G}[4] \oplus \mathtt{G}[3] \oplus \mathtt{G}[2]$ combination as if a limit of 1 was not in use, the algorithm would go on to compare $\mathtt{G}[4] \oplus \mathtt{G}[3] \oplus \mathtt{G}[2] \oplus \mathtt{G}[1]$.

4.2 Gaussian Elimination

As a limit was imposed on the computational complexity of the adapted OFGE algorithm, and as a result its accuracy, the investigation then proceeded to adapt the Gaussian Elimination algorithm that was mentioned in Chapter 2 to obtain *exact* simulation results more efficiently.

To adapt the original GE process mentioned in Listing 2.1 to make it more efficient, a series of optimisations were introduced, the main components of which are listed below:

- The OFGE concept of performing a GE process every time a new packet is received was amalgamated by keeping the state of variables `cod_matrix` and `recoverable_packets` constant between new packets being received. This meant that the adapted algorithm is not carrying out a complete GE process

```
% Remove the already known packets from the new coded packet
new_packet(recoverable_packets) = 0;

% If each of the packets included in the new coded packet were known then there is no point
% in continuing. Also, if the decoding matrix is already full rank there is no need to
% perform any more GE
if (sum(new_packet) > 0) && (sum(recoverable_packets) < num_source)
  % Append the new packet to tbe K x K partially decoded matrix
  cod_matrix = [cod_matrix ; new_packet];

  % For each position on the diagonal
  for arg_count = 1 : num_source

      % Start by assuming that there is already an element in this diagonal slot
      arg_is_in_diagonal = 1;

      % As a row-echelon matrix is being formed, Check if the current diagonal element
      % is equal to 0
      if (cod_matrix(arg_count,arg_count) == 0)
          arg_is_in_diagonal = 0;
          % If it is, then see if there are any other coded packets which contain the arg_count th
          % packet
          idxs_of_swap_rows = find(cod_matrix(:, arg_count));
          for row_count = 1 : numel(idxs_of_swap_rows)
            % If other coded packets do contain the arg_count packet, and the arg_count packet is
            % the first packet to appear in the coded packet, move that packet to this row
            if sum(cod_matrix(idxs_of_swap_rows(row_count), 1 : arg_count - 1)) == 0
                arg_is_in_diagonal = 1;
                % Swap the rows
                cod_matrix([idxs_of_swap_rows(row_count), arg_count], :) ...
                = cod_matrix([arg_count, idxs_of_swap_rows(row_count)], :);
                break;
            end
          end
      end

      if arg_is_in_diagonal
          % Now that there is a 1 in the correct position on the diagonal, XOR the current vector
          % with every other encoding vector to make sure that this vector is the only one with
          % this element on the diagonal
          idxs_of_competing_rows = find(cod_matrix(:, arg_count));
          for row_count = 1 : numel(idxs_of_competing_rows)
              if( idxs_of_competing_rows(row_count) ~= arg_count )
                  cod_matrix(idxs_of_competing_rows(row_count),:) = ...
                      xor(cod_matrix(idxs_of_competing_rows(row_count),:), ...
                      cod_matrix(arg_count,:));
              end
          end
      end
  end

  % Simple back substitution starting from the end of the diagonal upwards allows obtains which
  % packets are recoverable

  % However, if the coding_matrix is now full rank then every source packet is recoverable
  % there is no need to follow through with back substitution
  if (sum(diag(cod_matrix)) < num_source)
      for i = num_source : -1 : 1
          if sum(cod_matrix(i,:)) == 1
              recoverable_packets(cod_matrix(i,:) == 1) = 1;
              cod_matrix(:,cod_matrix(i,:) == 1) = 0;
          end
      end
  else
      recoverable_packets = true(1, num_source);
  end
  cod_matrix = cod_matrix(1:num_source, :);
end
```

Listing 4.2 Simple example of Binary Gaussian Elimination algorithm used to obtain the row-echelon form of the received encoding matrix.

over *all* of the received coding vectors *every* time a new packet is received, instead, it is performing a full GE process over a *partially solved* encoding matrix of maximum size $K + 1$ instead of N.

- The first line removes any previously recovered source packets from the newly received coding vector, which greatly reduces the amount of purposeless row operations that could be carried out.
- The `arg_is_in_diagonal` flag reduces the time wasted during the XORing phase, where it is asserted that there should only be one coding vector with a certain element on the diagonal, by bypassing this entire section of the code if there is no coding vector with the correct element to be placed on the diagonal.
- Finally, there are a couple of IF statements both before the main section of the code and before the back-substitution takes place, which reduce the amount of unnecessary operations by bypassing either the entire function or part of it, if the algorithm has already recovered either all of the source packets or all of the source packets that were present in the new coding vector.

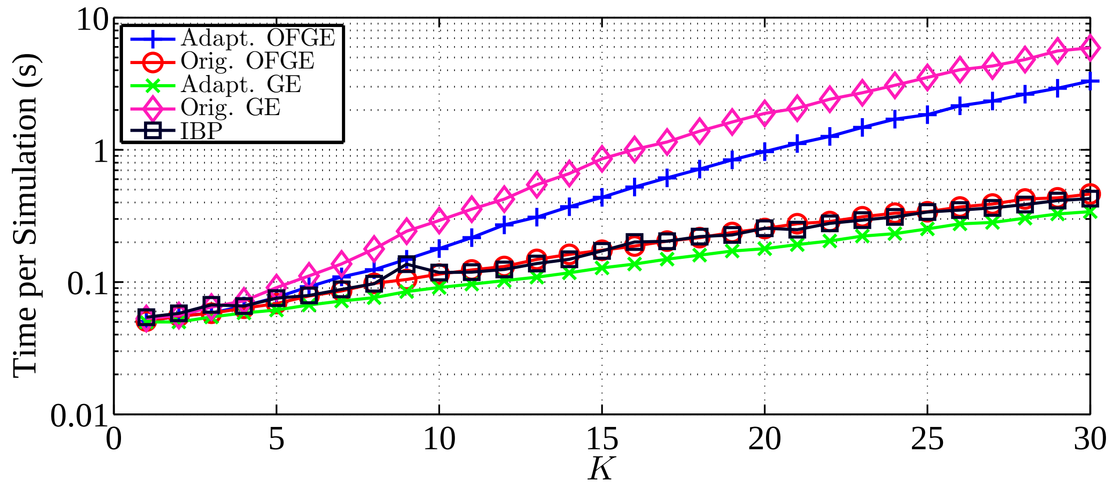
By optimising the original GE algorithm for MATLAB, and amalgamating the OFGE concept of performing a GE process whenever a coded packet it received, the investigation obtained an algorithm that executed faster and with greater accuracy than the adapted OFGE strategy.

4.3 Comparison of Decoding Schemes

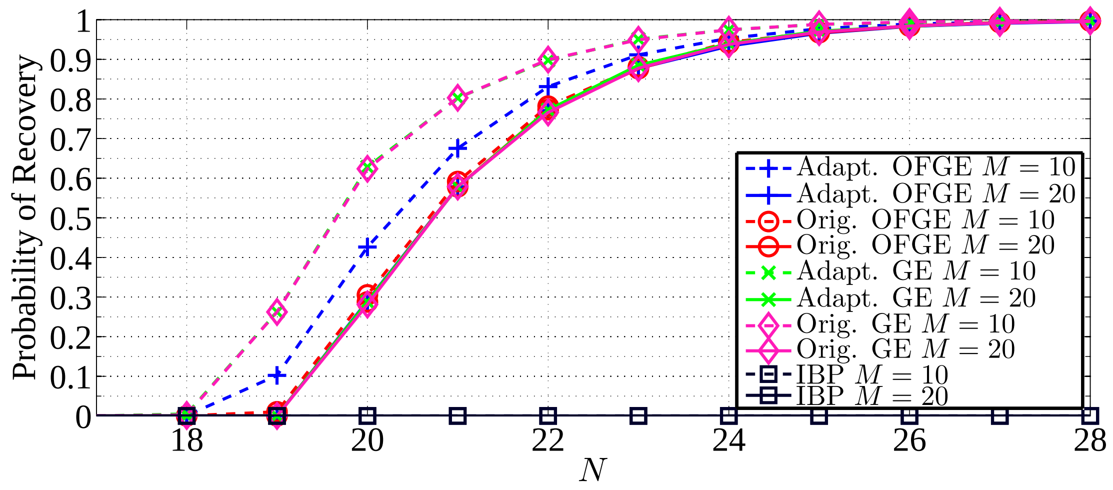
Figure 4.2 presents a comparison between the five different decoding strategies that were encountered during this investigation. These results were obtained on a platform equipped with a Intel i7 3770 processor and 8 GB of RAM.

As is mentioned above, and confirmed in the Figure, as the IBP scheme relies on a correctly configured degree distribution to function, its computational speed (close to Adapted GE and Original OFGE in Fig. 4.2 (a)) cannot outweigh its loss of performance when compared to the more complex algorithms that utilise Gaussian Elimination (Practically *no* progressive and overall performance seen in Fig. 4.2 (b)).

Also, although the adapted OFGE process has greater progressive performance that the original OFGE algorithm, as can be seen in Fig. 4.2 (a), the adapted GE scheme can recover source packets faster whilst exceeding the performance of the



(a) Time Comparison of each of the decoding schemes with differing source message lengths



(b) Performance Comparison of CFC encoding scheme with $K = 20, p = 0$

Fig. 4.2 Shows the difference in computational time for the decoding strategies (a) and the difference in performance when decoding rank-deficient coding matrices (b).

adapted OFGE algorithm. Because of this, the adapted GE decoding algorithm was chosen to be included in the simulation's system model.

4.4 Summary

In this chapter, the performance of both Iterative Belief Propagation and the Gaussian Elimination algorithm mentioned in Chapter 2 was compared, whilst introducing a new decoding strategy called On-the-fly Gaussian Elimination.

The next chapter will introduce a performance framework that will allow the interpretation of the two metrics from the transmitter's perspective. Simulation results for each of the encoding schemes will then contrasted and discussed.

Chapter 5

Performance Assessment of Layerless Schemes

In this chapter, performance curves will be presented for the previously mentioned encoding schemes. The majority of these results are simulated with the MATLAB platform, however, the results for certain schemes (SFC and OU) are given by the validated theoretical expressions described in Chapter 3. Firstly, a metric framework is introduced to allow interpretation of the set of results from a sender's perspective. Instead of simply examining the probability of recovering a portion or all of the source packets, the framework will start examining *how many* packets the sender needs to transmit to provide a certain probability of recovering a *portion* of the source message, and then how many *more* packets the sender needs to transmit to provide *full* recovery.

5.1 Metric Framework

The metric framework that will be used to interpret the two individual metrics, P_K and $\mathrm{P}_{K,M}$, will now be presented. Three new parameters are defined to forge links between the two separate metrics:

- $\hat{P}$, a predetermined target probability of packet recovery that a scheme has to attain.
- $\hat{N}$, the minimum number of transmitted packets that are needed to recover at least M source packets with a probability of at least $\hat{P}$.
- ΔN, the minimum number of additional packets that need to be transmitted to recover all K source packets with a probability of at least $\hat{P}$.

Figure 5.1 shows the progressive and overall response of the Ordered Uncoded scheme for a scenario with $K = 20$ and $p = 0.1$. If the target probability of packet

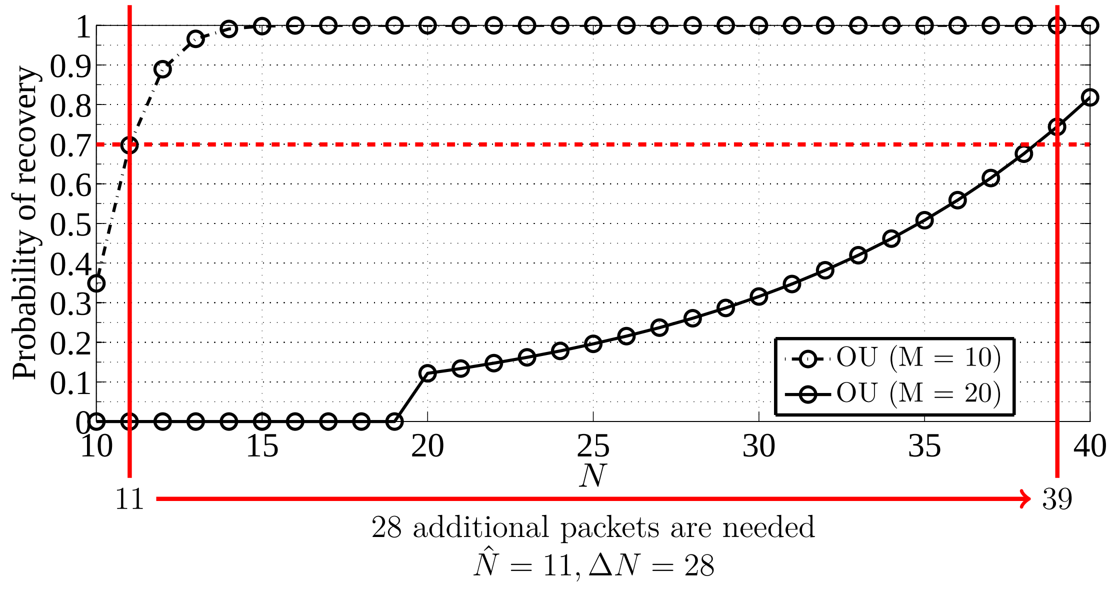


Fig. 5.1 Shows the values that $\hat{N}$ and ΔN would take for the OU scheme with $K = 20, p = 0.1$ and $\hat{P} = 0.7$.

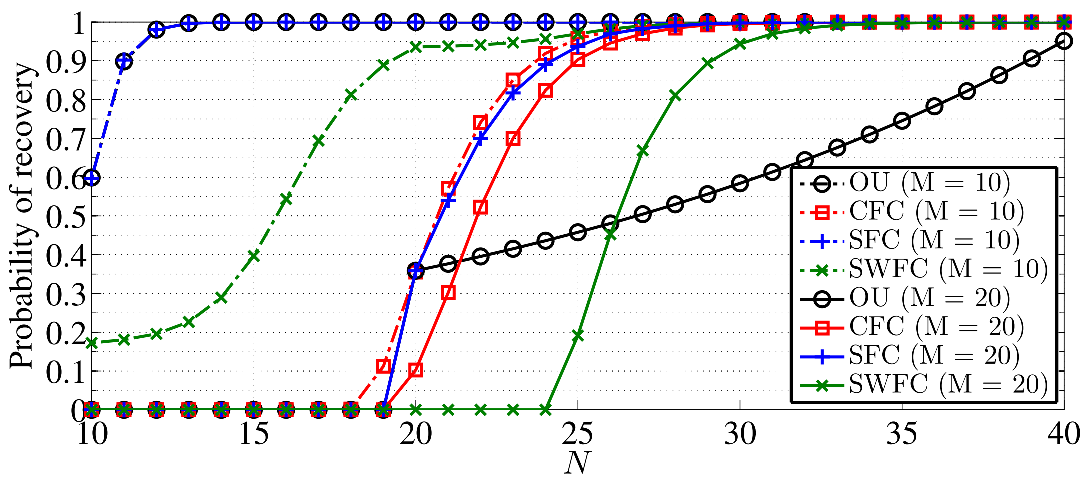
recovery is set to $\hat{P} = 0.7$ (depicted by the dashed red line), it is observed that the minimum number of transmitted packets needed to recover at least 10 source packets is $\hat{N} = 11$. Furthermore, the additional number of transmitted packets needed to recover the entire source message, with a probability of at least 0.7, is $\Delta N = 28$.

Although the OU scheme exhibits excellent progressive performance, confirmed by the low $\hat{N}$ value, the delay incurred in recovering the entire source message is notably large ($\Delta N > K$). The OU scheme's performance brings home the fact that the investigation should not be focussing on just lowering $\hat{N}$, but also minimising ΔN as well.

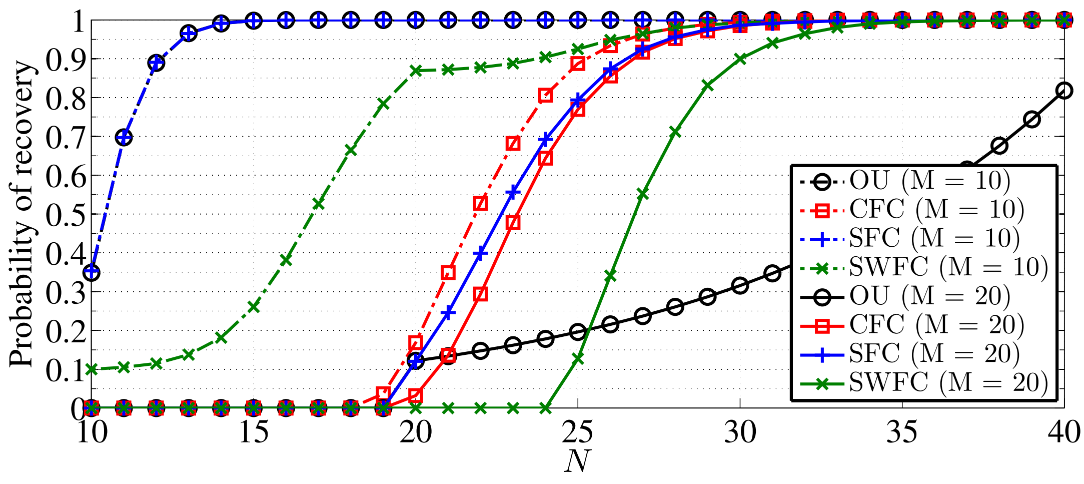
For example, if a user could quickly recover and start playing the first half of a source message but then ran out of media to play, they would have to wait to receive a set of packets that is, in total, nearly *double* the length of the entire source message. On the other hand, if a CFC had been utilised to broadcast the media, the user would have waited longer to start playing the media, but they would not have experienced any interruptions. These trade-offs will be explored in the following section.

5.2 Layerless Scheme Comparison

Simulated and theoretical results for the four encoding schemes introduced in Chapter 3 are now presented. The responses for a scenario with $K = 20$ and $p = \{0.05, 0.1\}$ are shown in Figure 5.2. Higher values of p *can* be used but it is



(a) $p = 0.05$



(b) $p = 0.1$

Fig. 5.2 Packet recovery probabilities as a function of N for $K = 20$.

assumed that the physical layer employs error correcting codes that improve the erasure probability as "seen" by the network/application layer, so that a maximum erasure probability of $p = 0.1$ is feasible.

Table 5.1 shows the values of $\hat{N}$ and ΔN for different erasure probabilities and the various schemes under investigation, where $\hat{P} = 0.9$. For example, for the $K = 40$, $p = 0.05$ case, it is observed that the OU scheme requires 22 packets to be transmitted in order to recover at least half of the source message with a probability of at least $\hat{P} = 0.9$. It can also be concluded that in order for the OU scheme to recover the entire source message with probability $\hat{P}$, 58 more transmissions are needed, which would lead to a unacceptably large delay.

It is apparent, for this scenario, that the SFC scheme is most appropriate. This is because the SFC strategy not only outperforms SWFC and CFC in terms of *progressive* packet recovery (with lower $\hat{N}$ values in Table 5.1), but also SWFC and OU if *full* packet recovery is considered (lower $\hat{N} + \Delta N$ values in Table 5.1).

Table 5.1 Values of $\hat{N}$ and ΔN for different erasure probabilities and the various schemes under investigation.

(a) $K = 20$

	OU		CFC		SFC		SWFC	
p	$\hat{N}$	ΔN	$\hat{N}$	ΔN	$\hat{N}$	ΔN	$\hat{N}$	ΔN
0.05	11	29	24	1	11	14	20	10
0.1	13	38	26	1	13	14	24	6

(b) $K = 40$

	OU		CFC		SFC		SWFC	
p	$\hat{N}$	ΔN	$\hat{N}$	ΔN	$\hat{N}$	ΔN	$\hat{N}$	ΔN
0.05	22	58	46	1	22	25	36	19
0.1	24	89	49	1	24	26	39	17

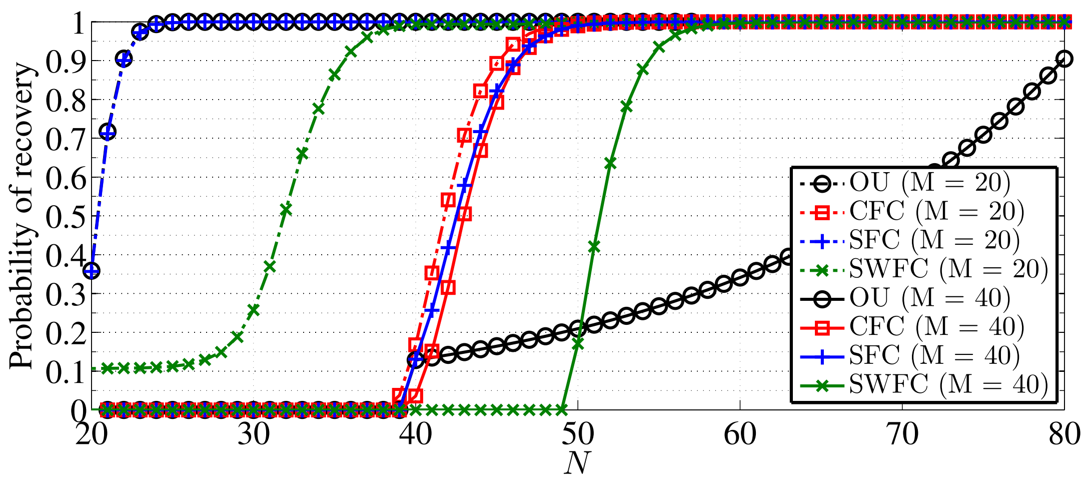
Considering the SWFC scheme, which combines $w = K/2$ source packets throughout the initial K transmissions, it can be noted that this scheme is more tolerant of the higher erasure probabilities. In other words, the progressive performance of this scheme degrades less than SFC and OU when the erasure probability is increased. For example, for $K = 20$ and $N = 10$, it can be seen in Figure 5.2 that the increase in erasure probability reduces $\mathrm{P}_{M,K}$ of SFC and OU by $\approx 25\%$ and $\mathrm{P}_{M,K}$ of SWFC by only $\approx 7\%$.

Similar trends can also be observed in Figure 5.3, for a larger source message size of $K = 40$. Note that the difference between $\mathrm{P}_{M,K}$ and P_K for the CFC scheme is negligible, as seen in both Figure 5.2 and Figure 5.3 as well as Table 5.1; this is because the CFC scheme makes no attempt to prioritise the recovery of the source packets that are closest to being utilised. On the other hand, the OU scheme exhibits excellent progressive performance, but as it is simply transmitting ordered source packets it is susceptible to an increase in the erasure probability of the channel. Note the steep increase in ΔN (depicted in Table 5.1) as the erasure probability is increased.

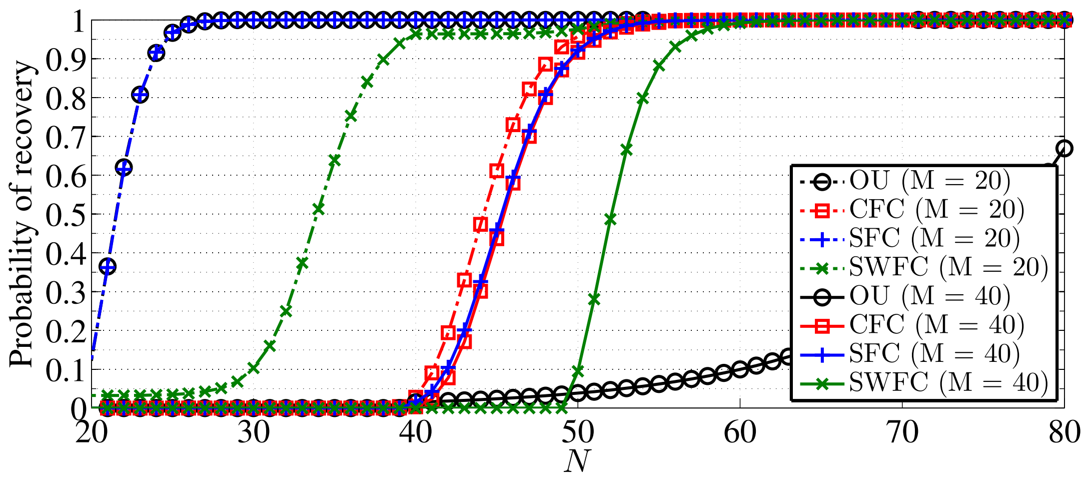
5.3 Summary

In this chapter, simulated and theoretical results for the encoding schemes that had been previously mentioned have been presented. A metric framework was then employed to interpret the two individual metrics, P_K and $\mathrm{P}_{K,M}$, from the transmitter's perspective.

Whereas, up to this point, the investigation has been concentrating on classless



(a) $p = 0.05$



(b) $p = 0.1$

Fig. 5.3 Packet recovery probabilities as a function of N for $K = 40$.

or layerless source messages, where each source packet has identical importance, the next chapter will introduce the concept of “layered” video encoding, where the source message is partitioned into a set of layers of *differing* importance.

The schemes presented in literature that attempt to adapt classical fountain codes, so that they may exhibit unequal-error-protection, and in particular, two methods called Non-Overlapping-Window Fountain Coding and Expanding-Window Fountain Coding are then touched on.

Next will be an assessment of whether the encapsulation of one the previously mentioned encoding schemes within these aforementioned windows provides any benefit in terms of performance. And finally, will be an examination of whether, from a progressive recovery point of view, adding Unequal Error Protection to a layered source message increases performance.

Chapter 6

Performance Assessment of Layered Schemes

In this chapter, the concept of a layered source message and some of the schemes presented in literature that try to tailor fountain codes for use with scalable video encoders are introduced. To examine the progressive and overall performance of these "layered" schemes, and to explore if their progressive performance can be enhanced at all, modifications to Chapter 3's encoding schemes are introduced. Simulation results are then presented and discussed for the modified encoding schemes.

Recently, the Joint Video Team of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) standardised a Scalable Video Coding (SVC) extension of the H.264 Advanced Video Coding (AVC) standard [13]. In previous chapters, the investigation has focussed on the examination of the performance of *layerless* encoding schemes, where every source packet has *identical* importance, to see how well receivers can progressively recover the media message. However, a video encoder that utilises the SVC extension produces multiple layers, each with a different level of importance.

The fountain codes that have been investigated up to this point provide Equal-Error-Protection (EEP) to each of the source packets, in that they do not alter the probability of a source packet's inclusion in a coded packet based on the source packet's importance. This method of operation was acceptable for the layerless scenario, but with the introduction of importance layers, fountain codes that exhibit Unequal-Error-Protection (UEP) are needed. This UEP approach is required to:

- Allow users to decode the most important layers before the full source block size is recovered (earlier recovery)
- Place more protection on the more important layers (reliable recovery)

Numerous methods of adapting fountain codes to provide UEP have been proposed in literature. Rahnavard *et al.* introduced, in [22], a method of providing UEP for a source message that has been partitioned into layers of different importance, by weighting the packets contained within each layer with a different probability of being selected, to be included in a coded packet, depending on their importance.

Windowed approaches have also found employment for providing UEP. The first scheme that established such a method, where a fountain encoder can only encode over the subset of the source message contained in a window, was presented by Studholme and Blake [23], known here as a Non-Overlapping Window (NOW) scheme. In a NOW scheme, each layer of a SVC encoded source message is assigned a probability of being selected that depends on its importance, with the most important layer usually having the largest chance of being selected. At each time step, the fountain encoder probabilistically selects a window to encode over and, since the most important layer is chosen to be included in coded packets more often, unequal error protection is obtained. The SWFC scheme that was mentioned in Chapter 4 could also be said to offer UEP in that the scheme prioritises the inclusion of source packets that are closest to being played.

The main difference between the windowed approaches described above and Rahnavard's weighted approach is that, in Rahnavard's approach, even though the probability of which source packets to select is biased towards including the more important layers, there is still a chance that *any* source packet could be included, regardless of importance layer. However, in windowed approaches, once the window has been selected, only source packets *within* the window can be included in coded packets.

An evolution of the windowing concept devised by Studholm and Blake is the Expanding-Window (EW) concept presented by Vukobratović *et al.* in [24] and expanded upon in [25]. In EW schemes, as opposed to NOW, the windows are not simply defined as the boundaries of each importance layer but as progressively increasing source block subsets which are aligned with the source's importance layers. This method, which will be expanded on in the following section, provides excellent protection to the most important layer as it is included in *every* coded packet.

6.1 Introduction to NOW and EW Strategies

The two main windowing schemes that aspire to provide UEP, the NOW and EW strategies, will now be introduced in more detail.

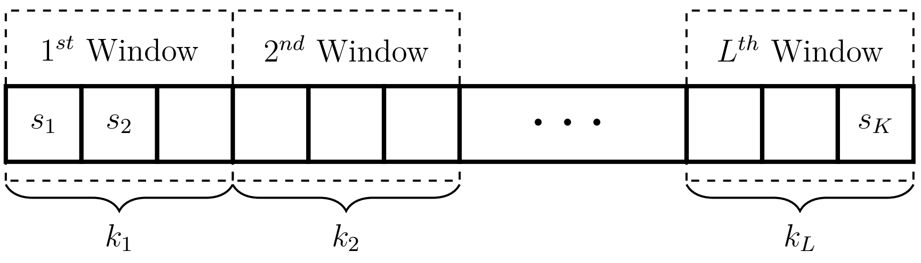


Fig. 6.1 An example of Non-Overlapping Window assignment.

6.1.1 Non-Overlapping Window

Let a layered source message $\mathbf{x}$ be comprised of K source packets in total that are partitioned into L importance layers. The number of source packets in the most important first layer is k_1, the number of packets in the second most important layer is k_2 and the number in the least important layer is k_L, so that $\sum_{i=1}^{L} k_i = K$.

The NOW scheme defines its windows, as mentioned above, to simply be the pre-defined importance layers already present in the source message. This behaviour can be seen in Figure 6.1. At each time step, the NOW encoder selects which window to generate packets over with the aid of a probability distribution Γ where the probability of choosing the first window is Γ_1, the second is Γ_2 and the L^{th} window is Γ_L such that $\sum_{i=1}^{L} \Gamma_i = 1$.

As with the encoding schemes that have been mentioned previously, the number of source packets that will be included in each coded packet, the degree distribution, is uniformly random.

6.1.2 Expanding Window

In contrast to the NOW scheme described above, the EW scheme defines its windows as progressively increasing source block subsets which are aligned with the source's importance layers, so that the c^{th} window's size is $k_{1:c} = \sum_{i=1}^{c} k_i$. This behaviour can be seen in Figure 6.2. In similarity with the NOW scheme, probability distribution Γ is again employed to select which window to encode over at each time step. It is palpable that the more important layers will be included in more coded packets, and hence, they are better protected.

A interesting point to note is that when the NOW scheme is implemented, coded packets that have been generated over a certain window can only be used for decoding a *single* specific layer. However, coded packets generated over the windows defined by the EW scheme can, in certain situations, assist in the decoding of *multiple* layers.

For example, coded packets that have been generated over the first window can assist in the decoding of the later windows. Also, if k_1 innovative packets that

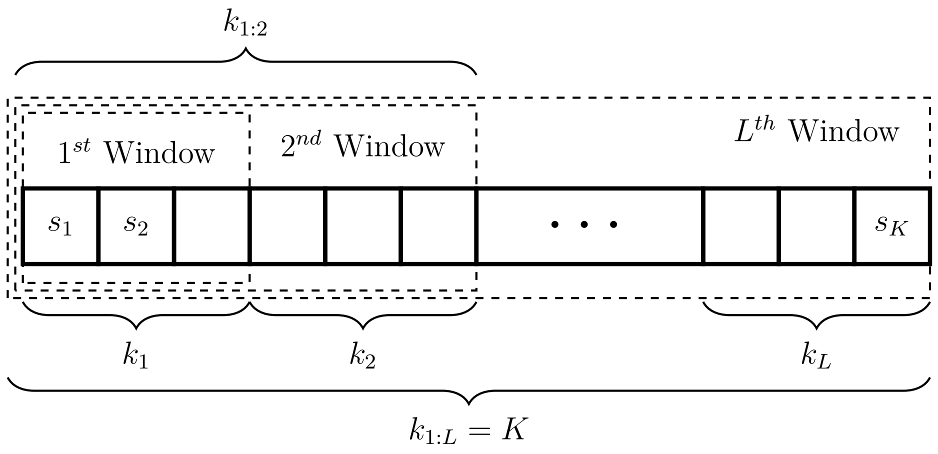


Fig. 6.2 An example of Expanding Window assignment.

have been encoded over the first window had not been received, although the first layer cannot be recovered on its *own* merit, it may be possible to recover the first window if a later window is successfully recovered, possibly with the assistance of the first window's coded packets.

Both the NOW and EW schemes usually utilise a CFC to generate coded packets within the selected window at a certain time step. As was seen in Figure 5.2 and 5.3, this decision trades progressive performance for overall performance. The investigation will now examine whether the progressive performance of these windowed strategies can be improved, and what cost this introduces. If a different method of encoding packets *within* a strategy's windows was selected, such as SFC, OU or SWFC, will this yield a benefit? The next section describes the operation of these encoding schemes within the NOW and EW framework.

6.2 Encapsulation of layerless encoding schemes

This section describes the operation of each of the four encoding schemes described in Chapter 3 *within* NOW and EW windows.

6.2.1 Conventional Fountain Code

Consider the generic layered source message with two importance layers depicted by Figure 6.3. The behaviour of a CFC within a NOW or EW can be described by adapting the definition of a CFC from (2.1).

For a NOW, coded packet t_n would be constructed at time step n, where $n = \{1, \ldots, N\}$ and N is the total number of transmitted coded packets, from the

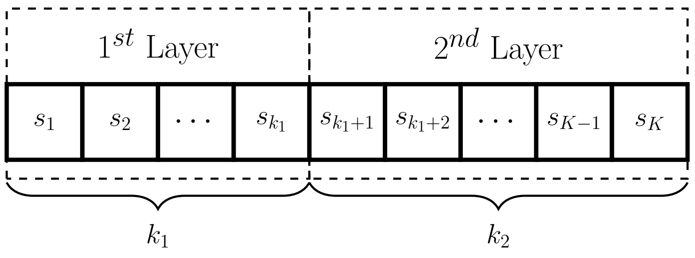


Fig. 6.3 A generic layered source message with two importance layers.

linear combination of source packets as follows

$$t_n = \sum_{i=k_{1:(c-1)}+1}^{k_{1:c}} g_{n,i}\, s_i \tag{6.1}$$

where $g_{n,i}$ is, again, a uniformly random binary coding coefficient, c is the probabilistically selected window during this time step, and, for ease of notation, $k_{1:0} = 0$.

For a EW, (6.1) simplifies to

$$t_n = \sum_{i=1}^{k_{1:c}} g_{n,i}\, s_i \tag{6.2}$$

6.2.2 Systematic Fountain Code

For a SFC if, at time step n, $n \leq k_c$ (NOW) or $n \leq k_{1:c}$ (EW) this code would transmit the n^{th} source packet within the selected window. Once this condition fails to be met for a *specific* window, the scheme will default to either (6.1) or (6.2) for this aforementioned window. To clarify, other larger windows may still transmit systematic packets.

6.2.3 Sliding Window Fountain Code

To define the behaviour of the SWFC within a NOW scheme, letting s_ℓ and s_r be the leftmost and rightmost source packets included in the window, the coded packet t_n, produced at time step n, can then be defined as

$$t_n = \sum_{i=\ell}^{r} g_{n,i}\, s_i\ . \tag{6.3}$$

where

$$\ell = k_{1:(c-1)} + \delta \left\lfloor \frac{n-1}{w} \right\rfloor + 1 \tag{6.4}$$

and $r = \ell + w - 1$. In order to simplify this investigation, let $w = \lceil k_c/2 \rceil$ and $\delta = \lceil k_c/4 \rceil$. Again, the SWFC scheme defaults to (6.1) immediately after $r \geq k_{1:c}$.

For the EW scenario, the behaviour of the SWFC scheme is identical to that described in Chapter 3, apart from the fact that $w = \lceil k_{1:c}/2 \rceil$ and $\delta = \lceil k_{1:c}/4 \rceil$ and that the scheme now defaults to (6.2) immediately after $r \geq k_{1:c}$.

6.2.4 Ordered Uncoded

If the benchmark OU scheme was deployed within a NOW, it would transmit source packet s_j at time step n where

$$j = \begin{cases} k_{1:c} & \text{if } \operatorname{mod}(n, k_c) = 0 \\ k_{1:(c-1)} + \operatorname{mod}(n, k_c) & \text{otherwise.} \end{cases} \tag{6.5}$$

Whereas the scheme simplifies to

$$j = \begin{cases} k_{1:c} & \text{if } \operatorname{mod}(n, k_{1:c}) = 0 \\ \operatorname{mod}(n, k_{1:c}) & \text{otherwise.} \end{cases} \tag{6.6}$$

when OU is included within EW.

To assess how well these windowed schemes, the original NOW and EW with a CFC and the NOW and EW schemes with different encoding strategies *within* the windows, can progressively recover the source message, a few modifications needed to be made to the progressive metric. Instead of considering a “single-layer” source message consisting of 40 packets with a single set of P_K and $\mathrm{P}_{K,M}$, a layered message with two layers of length 15 and 25 is now being examined, each layer with its own set of metrics.

For example, for the first importance layer, the investigation is interested in the probability of recovering the first half of *this* layer as quickly as possible P_{K,M,L_1}, and the probability of recovering the entire layer P_{K,L_1}. The probability of recovering *every* source packet in the entire message is also utilised, so that these layered results might be directly related to the layerless results.

The investigation is also interested in seeing how progressive performance is affected by not taking the layers into account when encoding, for example, by not windowing over the layers like EW and NOW. To clarify, the scheme that was found to be the most suitable during the layerless investigation (SFC) was applied to a layered source message (with $k_1 = 15$ and $k_2 = 25$) without prioritising any of the individual layers, so that the encoder just sees a “layerless” source message of 40 packets. The updated metrics for each *individual* layer that were described in the above couple of paragraphs are also applied. The next section shows the responses of these schemes over a layered source message.

6.3 Layered Scheme Comparison

The results presented in Figure 6.4 show the probability of recovering either a *portion* of a layer or the *entire* layer of a two-layer source message with $k_1 = 15$ and $k_2 = 25$. The probability of encoding over the first layer is slightly biased so that $\Gamma_1 = 0.6$, and consequently, the probability of the second window being selected is $\Gamma_2 = 0.4$. The probability of a specific packet being lost during transit over the lossy channel is $p = 0.05$.

6.3.1 Progressive Layer Recovery using NOW

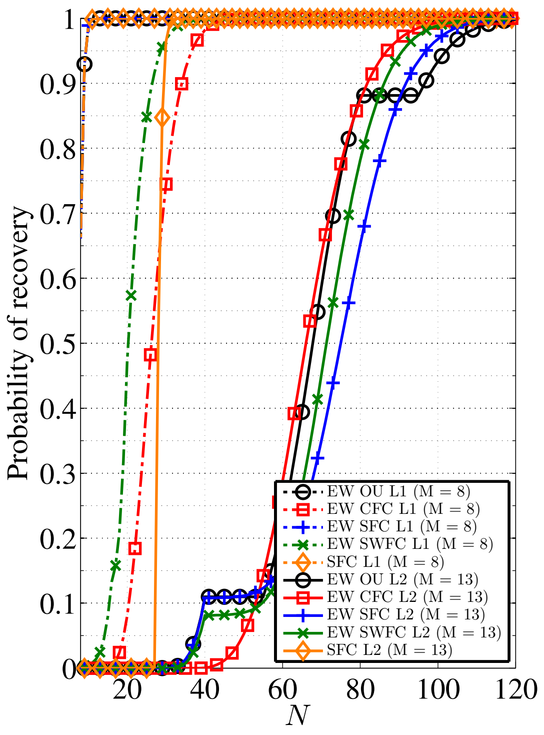
It is interesting to note how the change in the NOW SFC scheme from systematic to a CFC (Figure 6.4 (b)), with the addition of the requirement that a received coded packet is linearly independent, shifts the scheme's performance from the OU trend it was previously following.

It also makes sense that the layerless version of the SFC scheme has greater performance than the NOW version of SFC and OU, as these two schemes are encoding over *different layers* 40% of the time. As the NOW SWFC scheme switches to a CFC at $N = k_c$ it also makes sense that the scheme tracks the CFC response, especially if the second layer is considered, where there is a large probability (60%) of the encoder choosing the first layer to encode over, instead of the second.

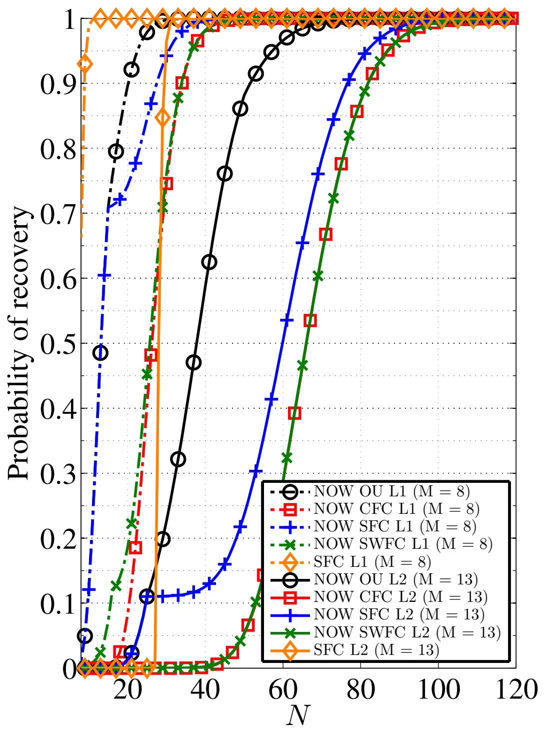
In similarity with the layerless results, the NOW OU scheme's response has the best progressive performance, when considering just the schemes within windows. However, the NOW OU scheme could be described as two smaller *layerless* messages each with a *much higher* erasure probability than the actual erasure probability, $p = 0.05$, of the channel in this scenario. This is, once again, because of the probability of the encoder picking the *other* window to encode over. It is logical then, that the NOW OU scheme's *overall* performance also follows the trend set by the OU scheme in the layerless assessment, in that it does not take kindly to large erasure probabilities. This trend can be seen in (Figure 6.4 (d)).

6.3.2 Progressive Layer Recovery using EW

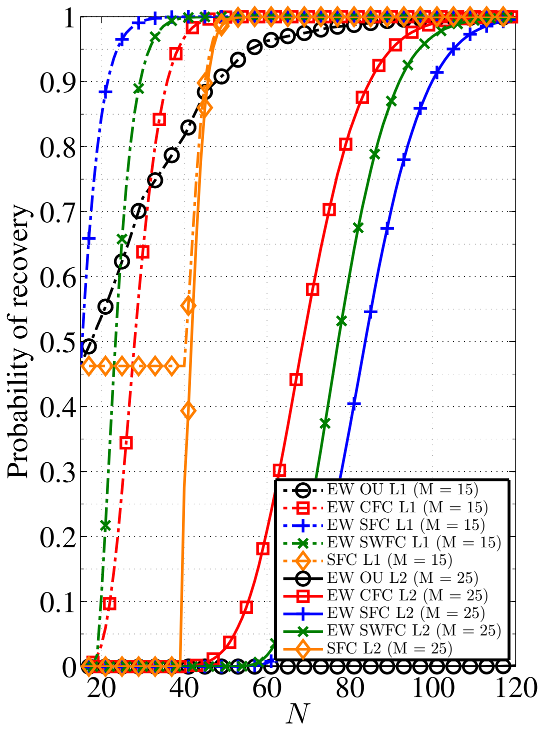
The performance of the encapsulated EW schemes presented in Figure 6.4 (a) is now considered. The EW version of OU and SFC now matches the layerless SFC scheme perfectly for the first layer. This is to be expected as, until $N > k_1$ all three schemes are identical. It is interesting to note the ceilings present in the NOW OU scheme's second layer response, at $N = \{40, 80\}$, that are the result of the first layer's $k_1 = 15$ packets being redundantly transmitted again.



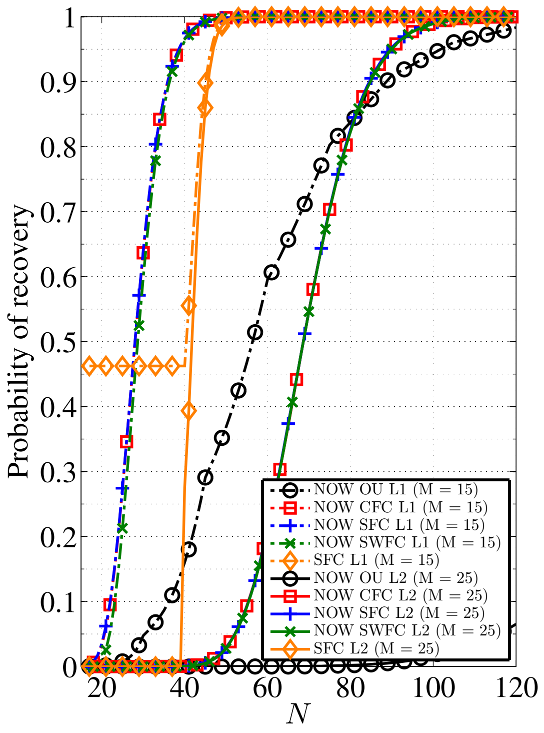
(a) EW $\mathrm{P}_{K,M}$ for individual layers



(b) NOW $\mathrm{P}_{K,M}$ for individual layers

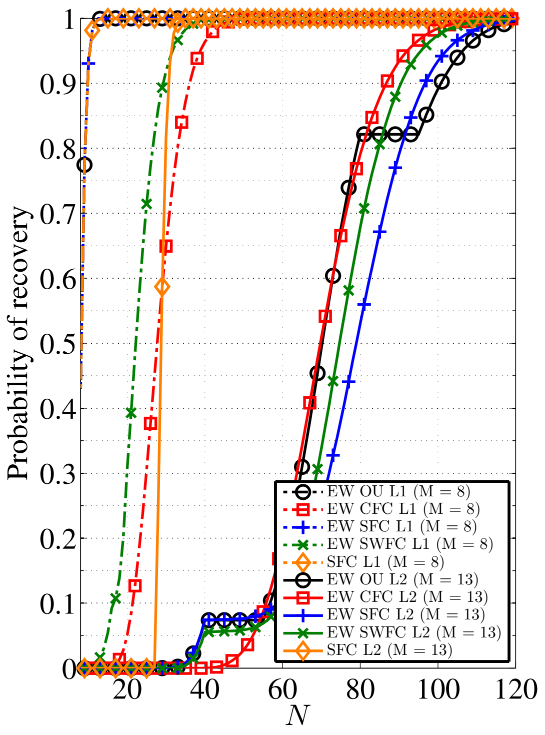


(c) EW P_K for individual layers

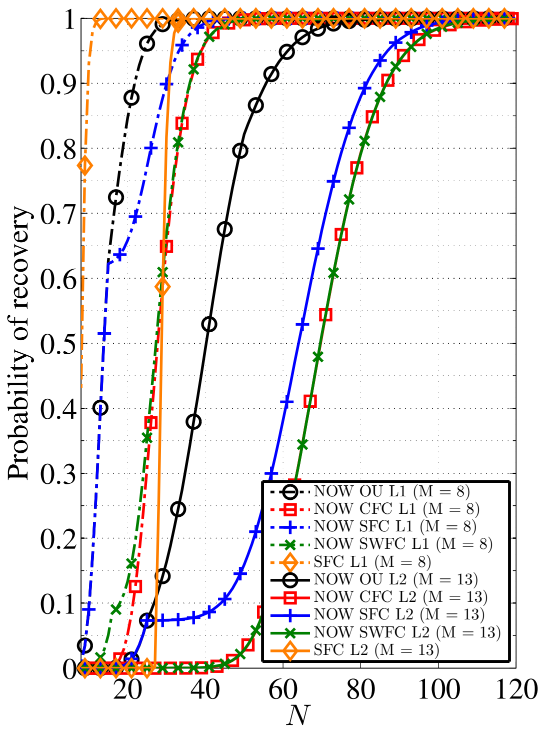


(d) NOW P_K for individual layers

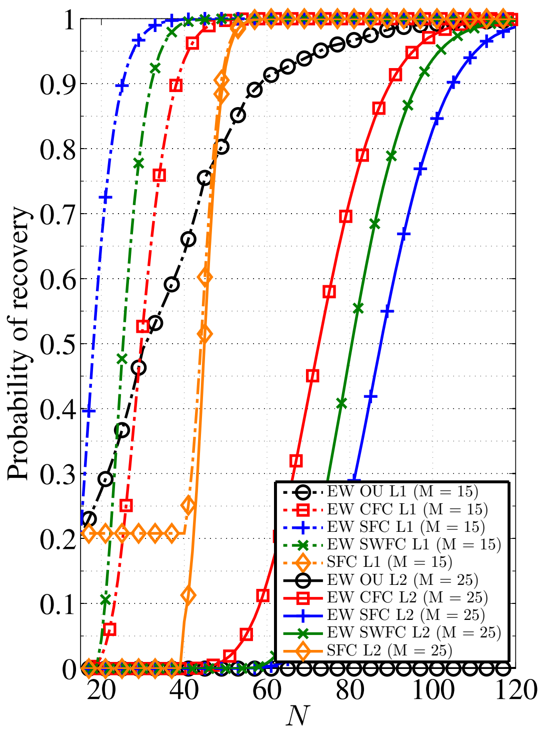
Fig. 6.4 Performance of the NOW and EW schemes with different encoding strategies within the windows for $K = \{15, 25\}$, $p = 0.05$. Layerless SFC has also been included in each plot.



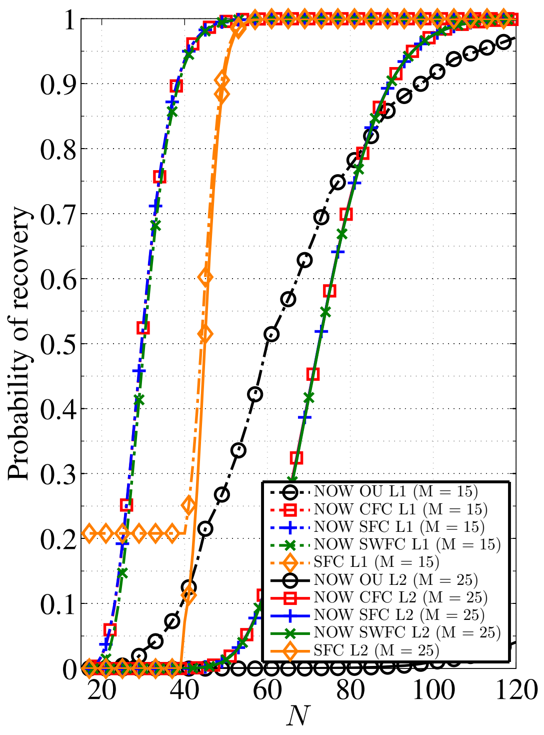
(a) EW $\mathrm{P}_{K,M}$ for individual layers



(b) NOW $\mathrm{P}_{K,M}$ for individual layers

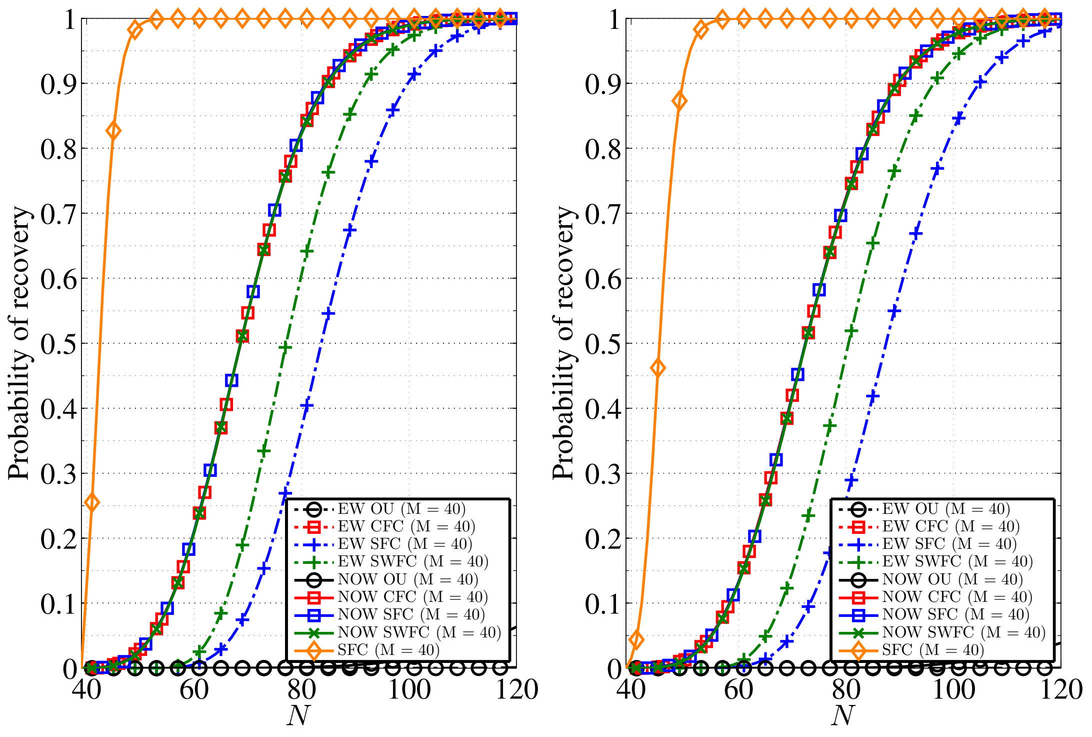


(c) EW P_K for individual layers



(d) NOW P_K for individual layers

Fig. 6.5 Performance of the NOW and EW schemes with different encoding strategies within the windows for $K = \{15, 25\}$, $p = 0.1$. Layerless SFC has also been included in each plot.



(a) $p = 0.05$ (b) $p = 0.1$

Fig. 6.6 The probability of recovering *every* packet in the source message for each of the NOW and EW schemes with different encoding strategies within the windows. $K = \{15, 25\}$ and $p = \{0.05, 0.1\}$. Layerless SFC has also been included in each plot.

Again, it is interesting to note the degradation in performance of EW SFC and EW SWFC as their second window swaps into a full CFC at $N = 40$, due to the introduction of coded packets making it more difficult to pick out the individual source packets that the progressive metric is looking for. The delay in the progressive recovery of the second layer when encoded with the layerless SFC stems from the fact that, the scheme needs to have transmitted at least $N = k_1 + \lceil k_2/2 \rceil$ systematic packets before there is even a small chance of recovering half of the second layer.

6.3.3 Entire Layer Recovery using NOW

The probability of fully recovering each source layer when the NOW scheme is used (Figure 6.4 (d)) is now examined. As mentioned above, it is observable that the NOW OU scheme does not perform at all well when confronted with what is equivalent to a large erasure probability, the probability of the encoder choosing the *other* window to encode over, with its response curves far to the right of the other examined schemes. A notable characteristic of this plot is that the performance of the NOW CFC, NOW SFC and NOW SWFC schemes appear to be identical. It is hypothesised that not receiving, on average, roughly *half* of the packets that a layer could generate, due to the other window being selected, wipes out the individual characteristics of the SFC and SWFC schemes so that when the two schemes switch to a CFC after k_c, their performance equates.

Another interesting point is the discontinuity in the first layer response of the NOW OU scheme at points where N is a multiple of k_1, when the first set of k_1 source packets are about to be retransmitted. The gradient of this response lessens as time passes, as it takes a large amount of transmissions (at least 15) to cycle through each iteration of k_1 packets, in order to have even a chance at recovering the specific packet(s) that have not yet been received.

The flat first layer response of the layerless SFC scheme makes sense as the decoder has to wait for the other $K - k_1$ packets to be transmitted before the scheme will switch to a full CFC to start filling in the source packets that have been lost through the channel.

6.3.4 Entire Layer Recovery using EW

Next, how well the EW schemes can fully recover each of source message layers (Figure 6.4 (c)) will be looked at. The EW SFC, EW OU and layerless SFC again start from the same point, but the EW SFC then switches to a CFC which really enhances its performance, whereas the EW OU scheme encounters the same sort of disjoint response that was described in the paragraphs above, with changes in response every k_1 transmitted packets.

It is also noteworthy that there is a trade-off between the performance of each *layer* for the EW SFC, EW CFC and EW SWFC schemes. The performance of the first layer directly relates to the reasoning given when examining the layerless results in Chapter 5, the second layer performance, however, is novel. It obviously relates to the *number* of linearly independent coded packets that have been encoded over the second layer. The EW CFC scheme produces the most, and obtains the best performance, as it has been encoding over the complete second window since the start of transmission. Following on, the EW SWFC produces more coded packets influenced by the second layer than the EW SFC scheme. One of the situations where this occurs is over the initial SWFC window, which is 20 packets long and therefore includes the first 5 second layer packets, these 5 packets then have the opportunity of being encoded over for 20 transmissions. The EW SFC scheme, on the other hand, does not have the opportunity to start sending systematic second layer packets until $N > k_1$. The trends described above also hold for a scenario with a larger erasure probability, such as $p = 0.1$. These results can be seen in Figure 6.5.

6.3.5 Entire Source Message Recovery using Both Schemes

If the results given in Figure 6.6 are also included, it could be concluded that since the layerless SFC scheme gives the best progressive performance and the best overall performance, in terms of recovering the *entire* source message, it is the best choice. However, if the objectives given towards the start of this chapter are reconsidered, it is observed that a priority in the design of these schemes must be the early *full* recovery of the most important layer, which would suggest the most suitable strategy would be one of the EW schemes.

6.4 Summary

In this chapter, the concept of a layered source message and the NOW and EW schemes that attempt to provide Unequal-Error-Protection to each of the source message's layers were introduced. The progressive and overall performance of these layered schemes were then introduced, and the effect of adding the OU, SFC, SWFC and CFC schemes *within* these windowed schemes on their progressive performance investigated. Finally, simulation results were presented and discussed for the considered layered encoding schemes.

Chapter 7

Conclusion

The main objective of this thesis was to examine the capability of fountain coded schemes, that were already present in literature, to provide progressive source message recovery. This task was accomplished with the comparison of the simulation results of numerous layerless schemes, four of which are noted in this document.

To facilitate the examination of the progressive capability of each scheme a new metric, $\mathrm{P}_{K,M}$, was introduced, along with a framework to summarise both the new metric and the probability of recovering the entire source message. Theoretical expressions for the benchmark (OU) scheme and the CFC and SFC schemes were derived.

So that accurate simulation results were obtained efficiently, three branches of decoding techniques were assessed for suitability. The adapted Gaussian Elimination algorithm depicted in Chapter 4 was chosen, as it was both exact and exhibited acceptable computational complexity.

After examination of the results in Chapter 5 it was apparent, for the channel conditions that were considered, that the SFC scheme was most appropriate, as it exhibited better progressive *and* overall performance.

Once the layerless scheme comparison had been completed, the next stage was to question whether the progressive performance of the windowing techniques used in the broadcast of layered source messages could be enhanced, with the introduction of encoding techniques *within* the windows.

A layerless version of the SFC scheme was also applied to the layered message to see if, from a progressive performance point of view, adding the extra encoding complexity associated with the windowing schemes is strictly necessary. After considering the results in Chapter 6 it can be observed that, if just progressive and overall performance is inspected, the layerless SFC scheme is clearly the most suitable candidate.

However, if one of the main purposes of adding Unequal-Error-Protection to layered source messages is reconsidered, the requirement that the most important layer should be ideally *fully* recovered *before* K coded packets have been transmitted, it can be concluded the SFC scheme does not meet this requirement. In light of this, an EW scheme would appear to be the most suitable alternative.

7.1 Individual Contribution

My individual contributions are summarised below:

- Derivation of theoretical expressions for both the probability of progressively recovering at least M source packets and the the probability of recovering the entire source message, when using a Systematic Fountain Code (Chapter 3).
- Development of a MATLAB simulation platform to obtain simulation results for the layerless and layered scenarios. This platform also allowed the use of a Robust Soliton distribution, as described in Chapter 2, to optimise decoding performance when using the IBP decoder.
- Adaptation of the original On-The-Fly Gaussian Elimination algorithm in an attempt to increase performance (Chapter 4).
- Adaptation of an exact Gaussian Elimination process to decrease its computational complexity (Chapter 4).
- Performed a thorough analysis and comprehensive discussion of simulation results (Chapters 5 and 6).

7.2 Areas for Future Work

In order to keep the theoretical derivations tractable, and the computational complexity of the decoder within the simulation platform small, this investigation focussed on examining fountain codes where the relationships between elements in coded packets were defined by a Galois Field of order 2, in other words by the XOR function. Where a Galois Field is simply a field that contains a finite number of elements, in our case 2, on which the operations of multiplication, addition, subtraction and division have been specifically defined [26].

For our uses, employing a higher order Galois Field means that the number of possible combinations of the source packets within each coded packet increases, so, in turn, the probability of the linear independence of a coded packet increases.

In fact, for a Galois Field of order larger than 2^8, the probability of linear independence tends to 1. Because of this, in most recent literature the authors use larger finite fields, for example [27]. It would be interesting to observe the effect of this increase on the progressive performance of the fountain coded schemes. However, our current theoretical expressions would not hold and this would require a redesign of our encoding and decoding algorithms.

During the early stages of this investigation, the Joint Scalable Video Model (JSVM) reference software [28] was examined, which is a reference tool used by the developers of the H.264 SVC codec. It is possible to use this software to generate scalable video files from normal video files and then, by using another tool within the system, extract and modify the individual packets within each layer.

The investigation experimented with removing half of the second layer and viewing the degradation in output, with mind to producing a demonstration. However, due to the time constraints imposed by the sheer learning curve in using the software, this did not materialise.

Another area that could be explored would be how the introduction of a degree distribution to the fountain coded schemes affects both the progressive performance and the number of row operations required when decoding.

Appendix A

Conference Paper: Performance Assessment of Fountain-coded Schemes for Progressive Packet Recovery

Attached is the paper we presented towards the end of July 2014 at the 9th International Symposium on Communications Systems, Networks and Digital Signal Processing in Manchester.

Performance Assessment of Fountain-coded Schemes for Progressive Packet Recovery

Andrew L. Jones, Ioannis Chatzigeorgiou and Andrea Tassi
School of Computing and Communications
Lancaster University, United Kingdom
Email: {a.jones2, i.chatzigeorgiou, a.tassi}@lancaster.ac.uk

***Abstract*—Fountain codes are gradually being incorporated into broadcast technologies, such as multimedia streaming for 4G mobile communications. In this paper, we investigate the capability of existing fountain-coded schemes to recover a fraction of the source data at an early stage and progressively retrieve the remaining source packets as additional coded packets arrive. To this end, we propose a practical Gaussian elimination decoder, which can recover source packets "on-the-fly". Furthermore, we introduce a framework for the assessment of progressive packet recovery, we carry out a performance comparison of the investigated schemes and we discuss the advantages and drawbacks of each scheme.**

***Keywords*—fountain coding; sliding window; Gaussian elimination; erasure channel; multicast communication.**

I. Introduction

Network layer protocols traditionally partition data into multiple packets and then repeatedly transmit them until they are successfully received. This approach requires the implementation of a feedback channel in which the receiver can request the retransmission of corrupted packets. Fountain codes, initially proposed in [1], do away with a dedicated feedback channel and ease wastefulness of resources by transmitting random linear combinations of source packets. The first practical implementation of fountain codes was LT codes [2] but it was not until the invention of Raptor codes [3] that the fountain principle found its way to recent standards, such as Long Term Evolution (LTE) [4] and Digital Video Broadcasting for Handheld devices (DVB-H) [5].

Even though fountain codes can be applied to a diverse set of reliability-focused applications, such as voice communications and data storage, they are not well suited to mission-critical or latency-intolerant applications. As randomness is an integral part of their design, there is no guarantee that source packets will be recovered in the correct order. Furthermore, the decoding process can only begin when a sufficiently large number of coded packets have been received; therefore, a receiver cannot obtain an early estimate of the source data and gradually refine them as additional packets are recovered.

Sliding-window fountain codes, which were proposed in [6] and extended in [7], incur a small performance penalty compared to "windowless" fountain codes, but address the issues of unordered packet recovery and limited memory storage at the receiving side. Nevertheless, the authors considered an erasure-free channel and a non-negative overhead, that is, decoding is initiated when the number of received coded packets is at least equal to the number of source packets.

The motivation for our work is to modify the on-the-fly Gaussian elimination decoder [8], so that source packets can be extracted from the pool of received coded packets as soon as possible. Partial recovery of the source data will provide an early insight into their information content. Full recovery will be progressively achieved as additional coded packets are received and added to the pool. As part of our objectives, we also propose and utilise a framework, which assesses the capability of schemes to progressively recover the source data for communication over erasure channels.

The remainder of this paper has been organised as follows. Section II describes the three schemes under investigation, namely conventional fountain coding, sliding-window fountain coding and systematic fountain coding. Section III presents the Gaussian elimination decoder, proposes a modification that allows source data to be progressively recovered and explains in detail the decoding process. The performance assessment framework and a simple uncoded transmission scheme, which will be used as a benchmark, are introduced in Section IV. Performance comparisons are presented and discussed in Section V whereas the main findings of the paper are summarised in Section VI.

II. Reviewed Fountain-coded Schemes

In this section, we describe the fountain-coded schemes under consideration. In all cases, a message comprising K source packets $s_1, s_2, \ldots, s_K$, is input to an encoder. The encoder generates N packets, $t_1, t_2, \ldots, t_N$, and transmits them over a broadcast erasure channel without feedback.

A. Conventional Fountain Coding

The encoder of a Conventional Fountain Code (CFC) constructs coded packet t_n at time step n, where $n = 1, \ldots, N$, from the linear combination or, equivalently, the bitwise sum of source packets as follows

$$t_n = \sum_{i=1}^{K} g_{n,i}\, s_i \tag{1}$$

where $g_{n,i}$ is a binary coding coefficient selected in an uniformly random manner. It is worth noting that we have chosen to employ a random distribution in order to examine the worst-case performance of fountain coding. In the rest of

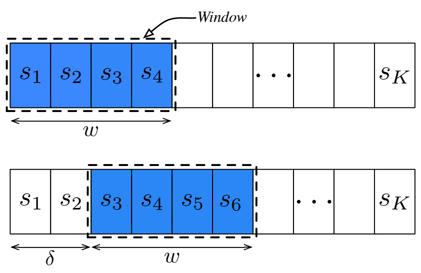

Window
s_1 s_2 s_3 s_4 $\cdots$ s_K
w
s_1 s_2 s_3 s_4 s_5 s_6 $\cdots$ s_K
δ
w

Fig. 1. Sliding window scheme as proposed in [6], [7]. In this representation, $w = 4$ and $\delta = 2$, so every window is encoded over for 4 transmissions.

the paper we impose that binary coefficients associated to a coded packet cannot all be simultaneously null.

B. Systematic Fountain Coding

The Systematic Fountain Code (SFC) combines both uncoded and coded packet transmissions. In particular, the SFC that we considered sequentially transmits each of the K source packets uncoded (referred to as *systematic packets*). As soon as every source packet has been transmitted once, the scheme behaves like a CFC. Using (1), the considered SFC encoder produces a stream of systematic/coded packets where the n-th transmitted packet can be defined as follows

$$t_n = \begin{cases} s_n & \text{if } n \leq K \\ \sum_{i=1}^{K} g_{n,i}\, s_i & \text{otherwise.} \end{cases} \tag{2}$$

C. Sliding Window Fountain Coding

The Sliding Window FC (SWFC) scheme considers a fixed window of size w that is moved along the source message by δ packets after w coded packets have been transmitted over each window, as shown in Fig. 1. As the sliding window is moving along the source message chronologically, it may be possible to recover a subset of the source message before fully recovering the whole source packet stream. In order to maximise the probability that at least M source packets are recovered as soon as possible, we set w and δ to M and $M/2$, respectively. The considered w value allows the SWFC scheme to initially resemble a miniature fountain code, as the first window will be encoded over for w transmissions.

Formally, let s_ℓ and s_r be the leftmost and rightmost source symbol encompassed by the window, respectively. As long as $r < K$, the n-th coded packet can be defined as follows

$$t_n = \sum_{i=\ell}^{r} g_{n,i}\, s_i \tag{3}$$

where

$$\ell \doteq \delta \left\lfloor \frac{n-1}{w} \right\rfloor + 1 \tag{4}$$

and $r \doteq \ell + w - 1$, where $\lfloor\cdot\rfloor$ denotes the integer part of a number. For $r = K$, we let the SWFC scheme default to CFC, thus the expression of t_n is provided by (1).

III. Decoding for Progressive Packet Recovery

In this paper we employ a customised implementation of the On-the-Fly Gaussian Elimination (OFGE) decoding process proposed by V. Bioglio *et al.* [8]. The OFGE process was chosen as it offers either an improved decoding time or a more accurate solution, when compared to other decoding algorithms for fountain coding such as iterative belief propagation [2] and incremental Gaussian elimination [9].

In its original version, the OFGE process waits for enough innovative coded packets (namely, K linearly independent coded packets) to produce a full rank upper triangular matrix so that every source packet could be recovered by an efficient back substitution phase. This partially conflicts with the objectives of the paper, as we aim to recover some source packets before K linearly independent packets have been received. To this end, our version of the OFGE algorithm, as presented in the rest of this section, is characterised by a *XORing phase* that leads to the recovery of a fraction of the source packets before the reception of K coded packets.

In order to describe the modified OFGE implementation, it is worthwhile to provide the following definitions:

- let $\mathbf{G}$ be a $K \times K$ matrix and $\mathbf{G}[t]$ be its t-th row;
- let $\mathbf{g}_i$ be the i-th received vector of coding coefficients;
- let us define the *degree* of $\mathbf{g}_i$ as the number of non-zero components of the vector;
- let $\mathcal{I}(\mathbf{g}_i)$ be the index of the leftmost vector component equal to 1 in $\mathbf{g}_i$.

The proposed OFGE algorithm consists of three phases:

1. Triangularisation Phase
   - (i) If $\mathbf{G}[\mathcal{I}(\mathbf{g}_i)]$ is empty then insert $\mathbf{g}_i$ into $\mathbf{G}[\mathcal{I}(\mathbf{g}_i)]$ and move to the back-substitution phase.
   - (ii) If the degree of $\mathbf{G}[\mathcal{I}(\mathbf{g}_i)]$ is greater than the degree of $\mathbf{g}_i$, swap the $\mathcal{I}(\mathbf{g}_i)$-th row with $\mathbf{g}_i$. Otherwise, replace $\mathbf{g}_i$ with $\mathbf{g}_i \oplus \mathbf{G}[\mathcal{I}(\mathbf{g}_i)]$.
   - (iii) If the degree of $\mathbf{g}_i$ is greater than 0 go back to (i). Otherwise, move to the back-substitution phase.
2. Back-Substitution Phase
   - (i) Define a temporary matrix $\mathbf{L}$ which is equal to $\mathbf{G}$.
   - (ii) For any a which goes from K to 1 perform the following steps:
     - (a) Set to 0 all the elements of the matrix $\mathbf{L}$ which belong to the j-th column (for any j such that s_j has been already recovered).
     - (b) If $\mathbf{L}[a]$ has a degree of 1 then the $\mathcal{I}(\mathbf{L}[a])$-th source packet is recovered.
3. XORing Phase
   - (i) For any c and d which go from K to 2 and $c-1$ to 1, respectively, perform the following steps:
     - (a) If $\mathbf{G}[c] \oplus \mathbf{G}[d]$ has a degree of 1, the packet $\mathcal{I}(\mathbf{G}[c] \oplus \mathbf{G}[d])$ is recovered. Otherwise, if $d > 1$, for any e that goes from $d-1$ to 1.
       - If $\mathbf{G}[c] \oplus \mathbf{G}[d] \oplus \mathbf{G}[e]$ has a degree of 1, the packet $\mathcal{I}(\mathbf{G}[c] \oplus \mathbf{G}[d] \oplus \mathbf{G}[e])$ is recovered.
   - (ii) Move to the back-substitution phase and exit.

To give an example of the progressive OFGE decoding process, if the first coded packet that we receive is associated with the coding vector $\mathbf{g}_1 = [0,0,1,1,1]$ (namely, it is a linear combination of s_3, s_4 and s_5), the OFGE process will place $\mathbf{g}_1$ straight into $\mathbf{G}[3]$, as $\mathbf{G}[3]$ is currently empty. If $\mathbf{g}_2 = [1,1,0,1,0]$ is then received, as $\mathcal{I}(\mathbf{g}_2) = 1$ and $\mathbf{G}[1]$ is empty, $\mathbf{g}_2$ will be inserted straight into $\mathbf{G}[1]$. At this point, $\mathbf{G}$ looks like the left side of the following relationship

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{Recv.\, \mathbf{g}_3} \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \quad (5)$$

Then, if $\mathbf{g}_3 = [1,0,1,0,1]$ then, as $\mathcal{I}(\mathbf{g}_3) = 1$ and $\mathbf{G}[1]$ already contains a coding vector of lesser or equal degree, $\mathbf{g}_3$ is replaced by $\mathbf{g}_3 \oplus \mathbf{G}[1] = [0,1,1,1,1]$. And as $\mathcal{I}(\mathbf{g}_3)$ is now 2 and $\mathbf{G}[2]$ is empty, the new value of $\mathbf{g}_3$ is inserted directly into $\mathbf{G}[2]$, namely, the right side of (5). During the XORing phase, the combination of $\mathbf{G}[3] \oplus \mathbf{G}[2] = [0,1,0,0,0]$ will be considered. Hence, source packet s_2 will be marked as recovered.

Let us imagine that $\mathbf{g}_4 = [0,1,1,0,0]$ is now received. As $\mathcal{I}(\mathbf{g}_4) = 2$ and $\mathbf{G}[2]$ contains a encoding vector of greater degree, $\mathbf{g}_4$ is swapped with $\mathbf{G}[2]$. As $\mathbf{g}_4$ has a greater degree than $\mathbf{G}[2]$, $\mathbf{g}_4$ is replaced by $\mathbf{g}_4 \oplus \mathbf{G}[2] = [0,0,0,1,1]$. Now $\mathcal{I}(\mathbf{g}_4) = 4$ and $\mathbf{G}[4]$ is empty, so $\mathbf{g}_4$ is inserted straight into $\mathbf{G}[4]$. As a consequence, $\mathbf{G}$ is equal to the left side of the following relationship

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{Recv.\, \mathbf{g}_5} \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \quad (6)$$

Matrix $\mathbf{G}$ is then passed to the back-substitution phase which, as s_2 has already been recovered, finds that $\mathbf{G}[2]$ has a degree of 1. Source packet s_3 is now marked as recovered. If the next received vector is $\mathbf{g}_5 = [0,0,1,0,1]$, it is immediately swapped with $\mathbf{G}[3]$ because of its lesser degree. Vector $\mathbf{g}_5$ now has a degree larger than that of $\mathbf{G}[3]$, so $\mathbf{g}_5$ is replaced by $\mathbf{g}_5 \oplus \mathbf{G}[3] = [0,0,0,1,0]$. Now $\mathcal{I}(\mathbf{g}_5) = 4$ and $\mathbf{G}[4]$ contains an encoding vector of greater degree, so $\mathbf{g}_5$ is swapped with $\mathbf{G}[4]$. The vector $\mathbf{g}_5$ now has a degree larger than that of $\mathbf{G}[4]$, so $\mathbf{g}_5$ is replaced by $\mathbf{g}_5 \oplus \mathbf{G}[4] = [0,0,0,0,1]$. As $\mathcal{I}(\mathbf{g}_5) = 5$ and $\mathbf{G}[5]$ is empty, $\mathbf{g}_5$ is inserted straight into $\mathbf{G}[5]$.

At this stage, $\mathbf{G}$ has assumed the form in the right side of (6); it is then passed to the back-substitution phase which finds rows of degree 1 whilst examining $\mathbf{G}[4]$ and $\mathbf{G}[5]$. Source packets s_4 and s_5 are now marked as recovered. As all instances of the recovered packets in matrix $\mathbf{L}$ have been set to 0, the back-substitution phase will also recover s_1.

IV. Performance Assessment Framework

Throughout this paper we consider packet transmission over a broadcast erasure channel without feedback. As it is often assumed, the probability p of a packet erasure captures the average quality of both the communication channel and the underlying error-correcting capability of the physical layer. In this section, we introduce a method for assessing the capability of a scheme to progressively recover packets. Prior to this, we define two useful performance metrics and compute them for ordered uncoded transmission. This will be used as a benchmark for the performance comparison of the aforementioned fountain-coded schemes.

A. Performance Metrics

We denote the probability that all K source packets have been successfully recovered at the destination, when $N \geq K$ packets have been transmitted, as $\mathrm{P}_K(N)$. This metric measures the capability of transmission schemes to recover the full source message as soon as a sufficient number of packets (at least K) has been broadcast.

On the other hand, $\mathrm{P}_{K,M}(N)$ shall signify the probability that *at least* M source packets from subset $\{s_1, \ldots, s_m\}$ have been recovered, given that packets $t_1, t_2, \ldots, t_N$ have been transmitted, where $M \leq m \leq \min(K, N)$. Metric $\mathrm{P}_{K,M}(N)$ measures the capability of a scheme to retrieve and possibly use – immediately after reception – a fraction of the source packets, which either precede or are contemporary with the last transmitted packet. For example, assume that a message comprises source packets $s_1, \ldots, s_{10}$ and the encoder transmits packets $t_1, \ldots, t_6$ in six time steps. The proposed metric focuses on the recovery of some or all source packets from subset $\{s_1, \ldots, s_6\}$, even if source packets that come after t_6 in time have been recovered at the destination.

B. Ordered Uncoded Transmission

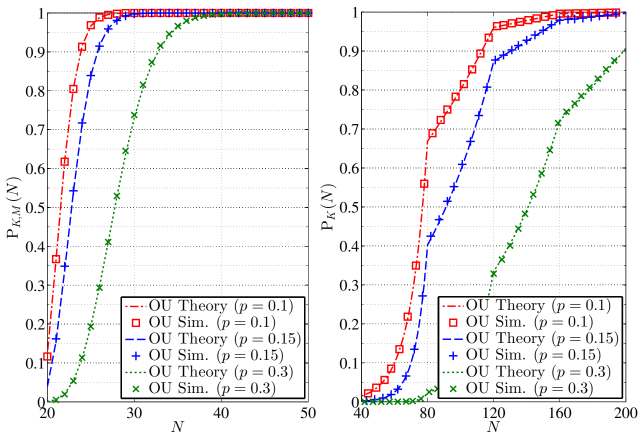
Having defined $\mathrm{P}_K(N)$ and $\mathrm{P}_{K,M}(N)$, we shall now obtain closed-form expressions for the case of Ordered Uncoded (OU) transmission, which will be used as a performance benchmark in our study. Note that the term *uncoded* transmission implies that the transmitted packets are not linear combinations of the source packets. In OU transmission, the K source packets are sequentially transmitted and periodically repeated. At time step $n = jK + i$, the transmitted packet is

$$t_{jK+i} = s_i, \text{ for } j \geq 0 \text{ and } i = 1, \ldots, K. \quad (7)$$

If the allocated transmission energy and time are sufficient to broadcast $N = \alpha K + \beta$ packets, where α, β are non-negative integers, we understand that $(\alpha+1)$ copies of packets $s_1, \ldots, s_\beta$ and α copies of packets $s_{\beta+1}, \ldots, s_K$ will be transmitted. The probability of recovering all source packets at the destination is the probability that at least one copy of each of the first β and the last $(K-\beta)$ source packets will be received, that is

$$\mathrm{P}_K(N) = \left(1 - p^{\alpha+1}\right)^{\beta} \left(1 - p^{\alpha}\right)^{K-\beta}. \quad (8)$$

Parameters α and β can be expressed in terms of N and K as $\alpha = \lfloor N/K \rfloor$ and $\beta = (N \bmod K)$, where mod denotes the modulo operation.



(a) $M = 20$ (b) $M = 40$

Fig. 2. Performance validation of OU transmission for $K = 40$, different values of p and (a) partial message recovery ($M = 20$) or (b) full message recovery ($M = 40$).

We now alter our focus from the recovery of the full set of source packets to the retrieval of a smaller set of m packets, where $M \leq m \leq \min(K, N)$. After some manipulation, we arrive at the following expression for the probability of recovering a specific instance of *exactly* m packets, provided that $h \geq 0$ of them are among the first β packets and the remaining $(m - h)$ occupy the last $(K - \beta)$ positions,

$$f(m, h) = \left(1 - p^{\alpha+1}\right)^h \left(1 - p^{\alpha}\right)^{m-h} p^{\alpha(K-m)+\beta-h}. \quad (9)$$

The probability of recovering *at least* M source packets can be obtained from (9) for all valid values of m and h, that is

$$\mathrm{P}_{K,M}(N) = \sum_{m=M}^{K} \sum_{h=h_{\min}}^{h_{\max}} \binom{\beta}{h} \binom{K-\beta}{m-h} f(m, h) \qquad (10)$$

where $h_{\min} = \max(0, m-K+\beta)$ and $h_{\max} = \min(\beta, m)$. We note that the upper limit on m can be relaxed from $\min(K, N)$ to K; in the event of $N \leq m < K$, $f(m, h)$ will be zero and all unnecessary terms in (10) will be eliminated.

Fig. 2 compares analytical results with simulation results for $K = 40$ source packets and different values of erasure probability p. We observe in Fig. 2a that expression (10) accurately determines $\mathrm{P}_{K,M}(N)$ which, in this example, corresponds to the probability of recovering at least half of the source packets ($M = 20$) in the correct order. Similarly, simulation results for $\mathrm{P}_K(N)$ are in agreement with the values obtained from (8), as shown in Fig. 2b. As expected, $\mathrm{P}_K(N)$ can be seen as a special case of $\mathrm{P}_{K,M}(N)$ for $M = K$.

C. Assessment of Progressive Packet Recovery

Let $\hat{P}$ be the predetermined target probability of packet recovery for a transmission scheme. In order to assess the capability of that scheme to progressively recover the source message of K packets, we use $\hat{N} \leq N$ to represent the minimum number of transmitted packets that are required for the recovery of at least M source packets with probability $\mathrm{P}_{K,M}(\hat{N}) \geq \hat{P}$. Furthermore, we denote as ΔN the minimum number of additional packets that need to be transmitted to recover all K source packets with probability $\mathrm{P}_K(\hat{N} + \Delta N) \geq \hat{P}$.

TABLE I
VALUES OF $\hat{N}$ AND ΔN FOR OU TRANSMISSION, $K = 40$, $M = 20$, $\hat{P} = 0.9$ AND DIFFERENT VALUES OF p.

p	$\hat{N}$	ΔN
0.10	24	89
0.15	26	105
0.30	33	166

For fixed values of K, M and $\hat{P}$, the smaller the value of $\hat{N}$ is, the faster the partial recovery of the source message will be. We also deduce that a small value of ΔN indicates that the transmission scheme under investigation needs only a few extra packets to make a transition from the partially retrieved message to the fully recovered message for the same probability $\hat{P}$.

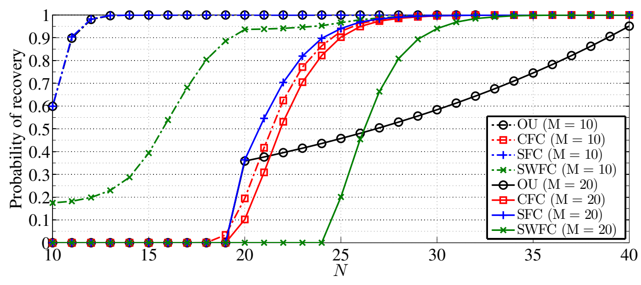
An example is given in Table I for OU transmission, $K = 40$, $M = K/2 = 20$ and $\hat{P} = 0.9$. The depicted values of $\hat{N}$ and ΔN generate probabilities $\mathrm{P}_{K,M}(\hat{N})$ and $\mathrm{P}_K(\hat{N} + \Delta N)$ that approach from above and are as close as possible to 0.9. They can both be obtained from Fig. 2 or derived from expressions (8) and (10). As shown in Table I, half or more of the source packets can be retrieved with probability 0.9 from a reasonably small number of transmitted packets $\hat{N}$. However, progressive packet recovery incurs a significant delay; we observe that the number of additional transmitted packets ΔN for the recovery of all source packets with probability 0.9 is markedly high. Note that ΔN increases considerably with an increase in the erasure probability p.

V. Results and Discussion

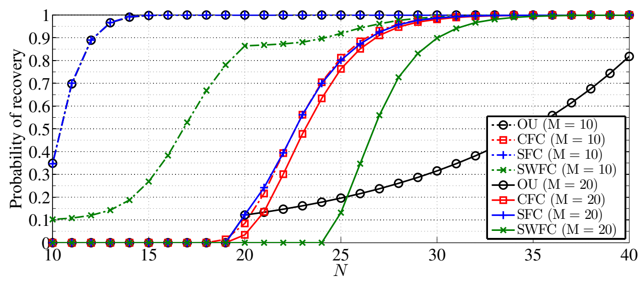
The responses of the reviewed schemes for $K = 20$ and $p = \{0.05, 0.1\}$, are shown in Fig. 3. It is apparent for this scenario that, with the aforementioned erasure channel, the SFC scheme is most appropriate. This is because SFC not only outperforms SWFC and CFC in terms of *progressive* packet recovery (lower $\hat{N}$ values in Table II), but also SWFC and OU if we consider *full* packet recovery (lower $\hat{N} + \Delta N$ values in Table II).

If we consider the SWFC scheme, which combines w source packets throughout the initial K transmissions, it can be noted that this scheme is more tolerant of the higher erasure probabilities. In other words, the progressive performance of this scheme degrades slower than SFC and OU when the erasure probability is increased. For example, for $K = 20$ and $N = 10$, it can be seen in Fig. 3 that the increase in erasure probability reduces $\mathrm{P}_{M,K}$ of SFC and OU by $\approx 25\%$ and $\mathrm{P}_{M,K}$ of SWFC by only $\approx 7\%$.

Similar trends can also be observed in Fig. 4, for $K = 40$. The capability of each reviewed scheme to progressively recover source packets has been summarised in Table II. Note

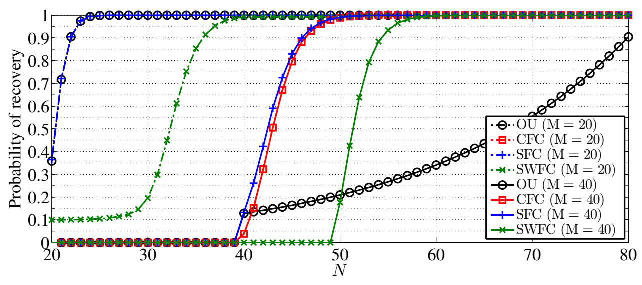


(a) $p = 0.05$

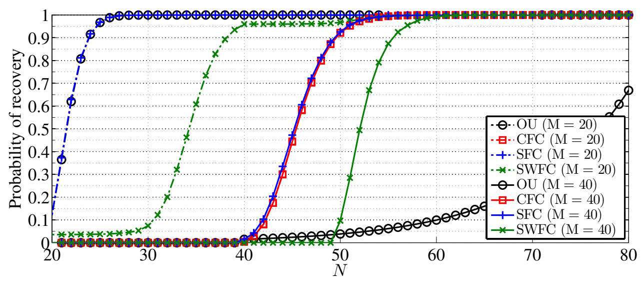


(b) $p = 0.1$

Fig. 3. Packet recovery probabilities as a function of N for $K = 20$.



(a) $p = 0.05$



(b) $p = 0.1$

Fig. 4. Packet recovery probabilities as a function of N for $K = 40$.

that the difference between $\mathrm{P}_{M,K}$ and P_K for the CFC scheme is negligible, if any, as seen in both Fig. 3 and Fig. 4 as well as Table II; this is because the CFC scheme makes no attempt to prioritise the recovery of the source packets which are closest to being utilised. On the other hand, the OU scheme exhibits excellent progressive performance, but as it is simply transmitting ordered source packets it is susceptible to an increase in the erasure probability of the channel. Note the steep increase in ΔN (depicted in Table II) as the erasure probability is increased.

Another interesting point, shown in both Fig. 3 and Fig. 4, is the change in the response of $\mathrm{P}_{M,K}$ for SWFC when the scheme defaults to CFC after K transmissions. The ceiling in the probability of recovery can be attributed to the fact that the first δ and the last δ source packets have had half the opportunities of being included in an coded packet, when compared to the other $K - w$ source packets. As K increases or p decreases, the ceiling tends to 1.

It is also interesting to note the tradeoff exhibited by SWFC, for a fixed window size w, between $\mathrm{P}_{M,K}$ and P_K when δ is altered. Although, for brevity, these results have been omitted. If $\delta < w/2$, the progressive recovery of SWFC is greatly enhanced, as each source packet will be included in a greater number of coded packets. However, as the encoding window is sliding much slower that previously, the time taken for the encoding process to have covered every source packet is substantially increased. This, of course, detrimentally affects P_K. The exact opposite reasoning holds for $\delta > w/2$.

VI. Conclusions

In this paper, we addressed the issue of progressive packet recovery in fountain-coded (FC) data transmission. We presented a novel extension of an efficient implementation of the Gaussian elimination algorithm known as the on-the-fly decoder. The considered FC schemes were assessed using a proposed framework and compared against ordered uncoded transmission. As expected, the FC-based schemes clearly outperform ordered uncoded transmission in terms of the probability of recovering the entire source message, regardless of the length of the message and the erasure probability. On the other hand, we established that the FC-based strategies require more transmission attempts than ordered uncoded transmission to recover a fraction of the source message. We also observed that the systematic FC scheme remarkably outperforms the other candidates in terms of progressive message recovery; it requires the smallest number of transmitted packets to retrieve at least half of the packets of the source message and progressively acquire the remaining packets. Nevertheless, if an

TABLE II
Values of $\hat{N}$ and ΔN for different erasure probabilities and the various schemes under investigation.

	OU		CFC		SFC		SW	
p	$\hat{N}$	ΔN	$\hat{N}$	ΔN	$\hat{N}$	ΔN	$\hat{N}$	ΔN
0.05	11	29	25	0	11	14	20	10
0.1	13	38	27	0	13	14	25	6

(a) $K = 20$

	OU		CFC		SFC		SW	
p	$\hat{N}$	ΔN	$\hat{N}$	ΔN	$\hat{N}$	ΔN	$\hat{N}$	ΔN
0.05	22	58	47	0	22	25	36	19
0.1	22	91	50	0	22	28	39	17

(b) $K = 40$

increase in overhead can be tolerated, the sliding-window FC scheme is an attractive alternative for systems using receiving equipment that can store only a limited number of packets.

Acknowledgment

This work was conducted as part of the R2D2 project, which is supported by the Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/L006251/1.

References

[1] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data," in *SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 4, Oct. 1998, pp. 56–67.

[2] M. Luby, "LT Codes," in *Proc. of the 43rd IEEE Symp. on Found. of Comput. Sci.*, Washington, DC, USA, 2002, pp. 271–280.

[3] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.

[4] M. Luby, M. Watson, T. Gasiba, T. Stockhammer, and W. Xu, "Raptor codes for reliable download delivery in wireless broadcast systems," in *3rd IEEE Consumer Commun. Networking Conference*, vol. 1, Jan. 2006, pp. 192–197.

[5] *Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Content Delivery Protocols*, ETSI Techn. Spec., Rev. 1.3.1, Jun. 2009.

[6] M. Bogino, P. Cataldi, M. Grangetto, E. Magli, and G. Olmo, "Sliding-Window Digital Fountain Codes for Streaming of Multimedia Contents," in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, May 2007, pp. 3467–3470.

[7] P. Cataldi, M. Grangetto, T. Tillo, E. Magli, and G. Olmo, "Sliding-Window Raptor Codes for Efficient Scalable Wireless Video Broadcasting With Unequal Loss Protection," *IEEE Trans. Image Process.*, vol. 19, no. 6, pp. 1491–1503, June 2010.

[8] V. Bioglio, M. Grangetto, R. Gaeta, and M. Sereno, "On the fly Gaussian elimination for LT codes," *IEEE Commun. Lett.*, vol. 13, no. 12, pp. 953–955, December 2009.

[9] S. Kim, K. Ko, and S. Chung, "Incremental Gaussian elimination decoding of raptor codes over BEC," *IEEE Commun. Lett.*, vol. 12, no. 4, pp. 307–309, April 2008.

Appendix B

Conference Paper: Binary Systematic Network Coding for Progressive Packet Decoding

Attached is the paper currently under review for the 2015 IEEE International Conference on Communications (ICC).

Binary Systematic Network Coding for Progressive Packet Decoding

Andrew L. Jones, Ioannis Chatzigeorgiou and Andrea Tassi
School of Computing and Communications, Lancaster University, United Kingdom
Email: {a.jones2, i.chatzigeorgiou, a.tassi}@lancaster.ac.uk

***Abstract*—We consider binary systematic network codes and investigate their capability of decoding a source message either in full or in part. We carry out a probability analysis, derive closed-form expressions for the decoding probability and show that systematic network coding outperforms conventional network coding. We also develop an algorithm based on Gaussian elimination that allows progressive decoding of source packets. Simulation results show that the proposed decoding algorithm can achieve the theoretical optimal performance. Furthermore, we demonstrate that systematic network codes equipped with the proposed algorithm are good candidates for progressive packet recovery owing to their overall decoding delay characteristics.**

***Keywords*—Network coding, Gaussian elimination, decoding probability, rank-deficient decoding.**

I. Introduction

Network coding (NC), originally proposed in [1], has the potential to significantly improve network reliability by mixing packets at a source node or at intermediate network nodes prior to transmission. The classical implementation of NC, which is often referred to as *straightforward NC* [2], randomly combines source packets using finite field arithmetic. As the size of the field increases, the likelihood of the transmitted packets being linearly independent also increases. However, the decoding process at the receiver is computationally expensive, especially if the field size is large. Furthermore, straightforward NC incurs a substantial decoding delay because source packets can be recovered at the receiver only if the received network-coded packets are at least as many as the source packets.

Heide *et al.* [3] proposed the adoption of binary systematic NC, which operates over a finite field of only two elements, as a means of reducing the decoding complexity of straightforward NC. A source node using systematic NC first transmits the original source packets and then broadcasts linear combinations of the source packets. The reduction in decoding complexity at the receiver decreases energy consumption and makes systematic NC suitable for energy-constrained devices, such as mobile phones and laptops. Lucani *et al.* [4] developed a Markov chain model to show that the decoding process of systematic NC in time division duplexing channels requires considerably fewer operations, on average, than that of straightforward NC. Barros *et al.* [5] and Prior and Rodrigues [6] observed that opting for systematic NC as opposed to straightforward NC reduces decoding delay without sacrificing throughput. Therefore, systematic network codes exhibit desirable characteristics for multimedia broadcasting and streaming applications. More recently, Saxena and Vázquez-Castro [7] discussed the advantages of systematic NC for transmission over satellite links.

As in [3], we also consider binary systematic network codes and investigate their potential in delivering services, such as multimedia and streaming, which often require the progressive recovery of source packets and the gradual refinement of the source message. Our objective is to prove that systematic NC not only exhibits a lower decoding complexity than straightforward NC, as shown in [4], but also a better performance, as observed in [5]. Even though our focus is on binary systematic NC, we explain that our analysis can be easily extended to finite fields of larger size. In addition, we develop a decoding algorithm and propose a framework, which helps us study the performance of systematic NC in terms of the probability of recovering a source message either in part or in full.

The rest of the paper has been organised as follows. Section II analyses the performance of systematic NC and introduces metrics for evaluating its capability of progressively recovering source messages. Section III proposes a modification to the Gaussian elimination algorithm that allows source packets to be progressively decoded. Section IV discusses the computational cost and accuracy of the proposed decoding algorithm, validates the derived theoretical expressions and contrasts the performance of systematic NC with that of benchmark transmission schemes. The main contributions of the paper are summarised in Section V.

II. Binary Systematic Network Coding

Let us consider a source node, which segments a message $\mathbf{s} = [\, s_i \,]_{i=1}^{K}$ into K source packets and encodes them using a systematic NC encoder. The encoder generates and transmits N packets, which comprise K systematic packets followed by $N - K$ coded packets. The systematic packets are identical to the source packets, while the coded packets are obtained by linearly combining source packets. The n-th transmitted packet, denoted by t_n, can be expressed as follows

$$t_n = \begin{cases} s_n & \text{if } n \leq K \\ \displaystyle\sum_{i=1}^{K} g_{n,i}\, s_i & \text{if } K < n \leq N \end{cases} \tag{1}$$

where $g_{n,i}$ is a binary coefficient chosen uniformly at random from the elements of the finite field GF(2). We can also express t_n in matrix notation as $t_n = \mathbf{G}[\,n\,] \cdot \mathbf{s}^{\mathsf{T}}$, where $\mathbf{G}[\,n\,] = [\, g_{n,i} \,]_{i=1}^{K}$ is the coding vector associated with t_n.

Note that when $n \leq K$, in line with the definition of binary systematic NC, we set $g_{n,i} = 1$ if $i = n$, else $g_{n,i} = 0$.

In the remainder of this section, we investigate the theoretical performance of systematic NC and derive analytical expressions for the probability of decoding the entire source message or a fraction of the source message. We also present performance metrics and benchmarks for the evaluation of systematic NC for progressive packet recovery.

A. Probability of Decoding the Entire Source Message

As previously mentioned, a source node using systematic NC transmits N packets, of which K are systematic and the remaining $N-K$ are coded. Assume that a receiver successfully recovers r packets, of which h are systematic and $r-h$ are coded. The coding vectors of the r received packets are stacked to form the $r \times K$ *decoding matrix* $\mathbf{G}$.

Let $f_K(r,N)$ denote the probability of decoding the K source packets given that r packets have been received. We understand that $f_K(r,N)$ is non-zero only if $K \leq r \leq N$. The value of r also determines the smallest allowable value of h. For instance, if $K \leq N < 2K$, the $N-K$ transmitted coded packets are fewer than the K transmitted systematic packets; given that $r \geq K$ packets are received, the number of received systematic packets h should be at least $r-(N-K)$. Otherwise, if $N \geq 2K$, the smallest value of h can be zero. Therefore, h is defined in the range $\max(0,\, r-N+K) \leq h \leq K$. Having defined the parameters of the system model and their interdependencies, we can now proceed with the derivation of an analytical expression for $f_K(r,N)$.

Lemma 1. *For $N \geq K$ transmitted packets, the probability of a receiver decoding all of the K source packets, given that $K \leq r \leq N$ packets have been successfully received, is*

$$f_K(r,N) = \frac{\binom{N-K}{r-K} + \sum_{h=h_{\min}}^{K-1} \binom{K}{h}\binom{N-K}{r-h} \prod_{j=0}^{K-h-1} \left(1 - 2^{-r+h+j}\right)}{\binom{N}{r}} \quad (2)$$

where $h_{\min} = \max(0,\, r-N+K)$.

Proof. The decoding probability $f_K(r,N)$ can be decomposed into the sum of the following probabilities

$$f_K(r,N) = \mathbb{P}\{h{=}K\} + \sum_{h=h_{\min}}^{K-1} \mathbb{P}\{h{<}K\}\, w_{K-h}(r-h). \quad (3)$$

The term $\mathbb{P}\{h = K\}$ represents the probability of recovering the K source packets directly from the K successfully received systematic packets. This is the case when $r-K$ out of the $N-K$ coded packets have been successfully delivered to the receiver along with the K systematic packets. Considering that r out of the N transmitted packets have been received, we can deduce that $\mathbb{P}\{h = K\}$ is given by

$$\mathbb{P}\{h{=}K\} = \frac{\binom{N-K}{r-K}}{\binom{N}{r}}. \quad (4)$$

The sum of products in (3) considers the probability of recovering $h < K$ systematic packets and decoding the remaining $K-h$ source packets from the $r-h$ received coded packets. More specifically, the probability $\mathbb{P}\{h < K\}$ of receiving h out of the K systematic packets and $r-h$ out of the $N-K$ coded packets is equal to

$$\mathbb{P}\{h < K\} = \frac{\binom{K}{h}\binom{N-K}{r-h}}{\binom{N}{r}}. \quad (5)$$

On the other hand, the probability of having $K-h$ linearly independent coded packets among the $r-h$ received ones can be obtained from the literature of straightforward NC, for example [8]. We find that

$$w_{K-h}(r-h) = \prod_{j=0}^{K-h-1} \left(1 - 2^{-(r-h)+j}\right). \quad (6)$$

Substituting (4), (5) and (6) into (3) gives (2). This concludes the proof. ☐

Proposition 1. *The probability of a receiver decoding all of the K source packets, after the transmission of $N \geq K$ packets over a channel characterized by a packet erasure probability p, can be expressed as follows*

$$\mathrm{P}_K(N) = \sum_{r=K}^{N} \binom{N}{r} (1-p)^r\, p^{N-r} f_K(r,N). \quad (7)$$

Proof. The proof follows from Lemma 1. The conditional probability $f_K(r,N)$ has been weighted by the probability of successfully receiving r out of N transmitted packets and averaged over all valid values of r. ☐

The closed-form expressions for the decoding probability of systematic network codes can be used to contrast their performance to the performance of straightforward network codes and give rise to the following proposition.

Proposition 2. *Systematic network codes exhibit a higher probability of decoding all of the K packets of a source message than straightforward network codes.*

Proof. For the same number of received packets r, the probability of decoding all of the K source packets is $f_K(r,N)$ for systematic NC and $w_K(r)$ for straightforward NC, where $w_K(r) = \prod_{j=0}^{K-1}\left(1 - 2^{-r+j}\right)$ as per (6). If we show that the relationship $f_K(r,N) \geq w_K(r)$ holds for all valid values of N, we can infer that the decoding probability of systematic NC is higher than that of straightforward NC. Dividing $f_K(r,N)$ by $w_K(r)$ gives

$$\frac{f_K(r,N)}{w_K(r)} = \binom{N}{r}^{-1}\left[\binom{N-K}{r-K}A + \sum_{h=h_{\min}}^{K-1} \binom{K}{h}\binom{N-K}{r-h} B_h\right] \quad (8)$$

where

$$A = \prod_{j=0}^{K-1} \frac{2^{r-j}}{2^{r-j}-1} \text{ and } B_h = \begin{cases} 1, & \text{for } h = 0 \\ \prod_{j=0}^{h-1} \frac{2^{r-j}}{2^{r-j}-1}, & \text{for } h > 0. \end{cases} \quad (9)$$

Note that $A > 1$ and $B_h \geq 1$ for all valid values of r, that is, $K \leq r \leq N$. Therefore, the right-hand side of (8) can become a lower bound on the ratio $f_K(r, N)/w_K(r)$ if coefficients A and B_h are removed. More specifically, we can obtain

$$\frac{f_K(r,N)}{w_K(r)} > \binom{N}{r}^{-1} \sum_{h=h_{\min}}^{K} \binom{K}{h}\binom{N-K}{r-h} \tag{10}$$

if the binomial coefficient $\binom{N-K}{r-K}$ in (8) is included into the sum and the upper limit of the sum is updated accordingly. We distinguish the following two cases for the value of N:

- $N \geq 2K$: In this case, we have $h_{\min} = 0$. Invoking a special instance of the Chu-Vandermonde identity [9, p. 41], we can reduce the sum at the right-hand side of (10) to
$$\sum_{h=0}^{K} \binom{K}{h}\binom{N-K}{r-h} = \binom{N}{r}. \tag{11}$$
- $K \leq N < 2K$: As previously explained, $h_{\min} = r-N+K$. Setting $h' = N-K-r+h$, expressing the sum in (10) in terms of h', exploiting the properties of binomial coefficients and using the widely-known Vandermonde's convolution [10, p. 29] gives
$$\begin{aligned}\sum_{h=r-N+K}^{K} \binom{K}{h}\binom{N-K}{r-h} &= \sum_{h'=0}^{N-r} \binom{K}{N-r-h'}\binom{N-K}{h'} \\ &= \binom{N}{N-r} = \binom{N}{r}.\end{aligned} \tag{12}$$

If we combine identities (11) and (12) with inequality (10), we obtain $f_K(r, N)/w_K(r) > 1$ for all valid values of N, which concludes the proof. We note that the ratio $f_K(r, N)/w_K(r)$ approaches 1 as the value of $N - K$ increases. □

Remark. Even though this paper is concerned with binary systematic NC, i.e. the elements of matrix $\mathbf{G}$ are selected uniformly at random from GF(2), the same reasoning can be employed to obtain $\mathrm{P}_K(N)$ when operations are performed over GF(q) for $q \geq 2$. The probability $f_K(r, N)$ of decoding the entire source message, given that r packets have been received, can be written as

$$f_K(r,N) = \frac{\binom{N-K}{r-K} + \sum_{h=h_{\min}}^{K-1} \binom{K}{h}\binom{N-K}{r-h} \prod_{j=0}^{K-h-1}\left(1-q^{-r+h+j}\right)}{\binom{N}{r}}. \tag{13}$$

Both Propositions 1 and 2 hold for $q \geq 2$. Substituting (13) into (7) gives the general expression for $\mathrm{P}_K(N)$.

B. Probability of Decoding a Fraction of the Source Message

In Section II-A, we focused on deriving the probability of decoding the K source packets when $N \geq K$ packets have been transmitted. Of equal interest is the probability of recovering at least $M < K$ source packets when $N \geq M$ packets have been transmitted. To the best of our knowledge, a closed-form expression for this probability, denoted hereafter as $\mathrm{P}_{K,M}(N)$, has not been obtained for straightforward NC. However, a good approximation, which follows readily from Proposition 1, can be computed for the case of systematic NC.

Corollary 1. *The probability of recovering at least $M < K$ source packets, when $N \geq M$ packets have been transmitted over a channel with packet erasure probability p, can be approximated by*

$$\mathrm{P}_{K,M}(N) \approx \sum_{r=M}^{N_{\min}} \binom{N_{\min}}{r} (1-p)^r \, p^{N_{\min}-r} \tag{14}$$

where $N_{\min} = \min(K, N)$.

Proof. The number of transmitted systematic packets is either N if $N < K$, or K if $N \geq K$. In general, $\min(K, N)$ systematic packets are sent over the packet erasure channel, for any value of N. If we wish to recover at least $M < \min(K, N)$ source packets and the erasure probability p is small, M or more received packets will most likely be systematic and, thus, linearly independent. As a result, the probability of decoding at least M source packets reduces to the probability of recovering at least M systematic packets, given by (14). □

We remark that the assumption of a low value of p is reasonable when the physical layer employs error correcting codes that improve the channel conditions as "seen" by higher network layers, where NC is usually applied. For example, the Long Term Evolution Advanced (LTE-A) framework considers an erasure probability of $p = 0.1$ [11].

C. Performance Metrics and Benchmarks

In order to assess the performance of systematic NC and explore its capability to progressively decode a source message, we will compare it with *ordered uncoded* (OU) transmission [12] and straightforward NC. In OU transmission, the K source packets are periodically repeated. The transmitted packet at time step $n = i + mK$ can be expressed as $t_{i+mK} = s_i$ for $i = 1, \ldots, K$ and $m \geq 0$. We note that transmission is *uncoded* in the sense that transmitted packets are not linear combinations of the source packets. By contrast, the n-th transmitted packet in straightforward NC is given by $t_n = \sum_{i=1}^{K} g_{n,i}\, s_i$ for $n > 0$, implying that all transmitted packets are linear combinations of the source packets.

Probabilities $\mathrm{P}_{K,M}(N)$ and $\mathrm{P}_K(N)$ will be used to contrast the performance of systematic NC, straightforward NC and OU transmission. In order to create links between the two decoding probabilities, we introduce the following parameters:

- $\hat{P}$ is a predetermined target probability of packet recovery that a transmission scheme has to attain. Probabilities $\mathrm{P}_{K,M}(N_1)$ and $\mathrm{P}_K(N_2)$ can be set equal to $\hat{P}$ in order to determine the number of transmitted packets N_1 and N_2 that are required for the partial or full recovery of the source message, respectively.
- $\hat{N}$ signifies the minimum number of transmitted packets required by the receiver to recover at least M source packets with a probability of at least $\hat{P}$.
- ΔN denotes the minimum number of additional packets that should be transmitted so that the receiver recovers the K source packets with a probability of at least $\hat{P}$.

A performance comparison of the investigated schemes will be carried out in Section IV. Prior to that, we discuss decoding algorithms for NC schemes and propose a decoding process that allows progressive decoding of source packets in the following section.

III. Progressive Decoding

If the objective of the decoding algorithm is the recovery of the K source packets after the reception of at least K transmitted packets, Gaussian Elimination (GE) could be used especially when the value of K is small. The GE algorithm transforms the decoding matrix $\mathbf{G}$ into row-echelon form [13]. The rank of the transformed matrix, which is equal to the rank of the original decoding matrix, can be obtained by inspecting the number of non-zero rows within the echelon form. If the rank is K, that is, if $\mathbf{G}$ is a *full-rank* matrix, the K source packets can be successfully recovered.

GE and schemes based on Belief Propagation (BP) [14] experience a large spike in computation when K transmitted packets are received. On-the-Fly Gaussian Elimination (OFGE) [15] manages to mitigate the decoding delay and computational complexity of GE by invoking an optimized triangulation process *every time* a packet is received. The OFGE decoder spreads computation out over each packet arrival and the decoding matrix $\mathbf{G}$ is already in partial triangular form by the time the K-th transmitted packet is received.

Both GE and OFGE have been designed to perform full-rank decoding. As a result, if the rank of $\mathbf{G}$ is less than K, that is, if the decoding matrix is *rank-deficient*, some source packets might still be decodable but GE or OFGE will not necessarily identify them. A modified version of OFGE, which we refer to as OFGE for Progressive Decoding (OFGE-PD), was presented in [12]. Similarly to OFGE, OFGE-PD also comprises a triangulation stage and a back-substitution stage. An additional stage, called the XORing phase, enables OFGE-PD to decode source packets from rank-deficient decoding matrices at the expense of increased computational complexity.

We revisited the original GE algorithm and we amalgamated the OFGE principle of initiating the decoding process whenever a packet is received. A sketch of the proposed algorithm, referred to as Gaussian Elimination for Progressive Decoding (GE-PD), is presented in Algorithm 1. To facilitate the description of GE-PD, we introduced function `Degree`, which determines the number of non-zero elements in a row vector; function `Diag`, which generates a row vector containing the elements of the main diagonal of a matrix; function `LeftmostOne`, which returns the position of the first non-zero entry in a row vector; and function `Swap`, which swaps two rows in a matrix. The decoding matrix $\mathbf{G}$ is initially set equal to the $K \times K$ zero matrix. Recall that $\mathbf{G}[n]$ represents the n-th row of $\mathbf{G}$, while $\mathbf{G}[i][j]$ denotes the entry of $\mathbf{G}$ in the i-th row and j-th column (equivalent to $g_{i,j}$). We note that, depending on the adopted programming language, the code can be further optimized and the execution speed of GE-PD improved.

Algorithm 1 Gaussian Elimination for Progressive Decoding

```
 1: Receive new $1 \times K$ coding vector $\mathbf{R}$
 2: Set entries in $\mathbf{R}$ that correspond to decoded packets to 0
 3: if (Degree($\mathbf{R}$) $> 0$) then
 4:     $\mathbf{G}[K+1] \leftarrow \mathbf{R}$
 5:     for $i = 1$ to $K$ do
 6:         $one\_in\_diag \leftarrow$ TRUE
 7:         if ($\mathbf{G}[i][i] = 0$) then
 8:             $one\_in\_diag \leftarrow$ FALSE,  $j \leftarrow i + 1$
 9:             repeat
10:                 if LeftmostOne($\mathbf{G}[j]$) $= i$ then
11:                     Swap($\mathbf{G}[i], \mathbf{G}[j]$)
12:                     $one\_in\_diag \leftarrow$ TRUE
13:                 end if
14:                 $j \leftarrow j + 1$
15:             until ( $j > K + 1$ ) or $one\_in\_diag$
16:         end if
17:         if $one\_in\_diag$ then
18:             for $j = 1$ to $(K + 1)$ do
19:                 if ( $j \neq i$ ) and ($\mathbf{G}[j][i] = 1$) then
20:                     $\mathbf{G}[j] \leftarrow \mathbf{G}[j] \oplus \mathbf{G}[i]$
21:                 end if
22:             end for
23:         end if
24:     end for
25:     $\mathbf{G} \leftarrow$ BackSubstitution($\mathbf{G}$, $K$)
26:     $\mathbf{G} \leftarrow$ Top $K$ rows of $\mathbf{G}$
27: end if
```

Function BackSubstitution($\mathbf{G}$, K)

```
1: for $i = K$ to 1 step $-1$ do
2:     if Degree($\mathbf{G}[i]$) $= 1$ then
3:         $j =$ LeftmostOne($\mathbf{G}[i]$)
4:         for $k = 1$ to $K$ do
5:             if $k \neq i$ then $\mathbf{G}[k][j] \leftarrow 0$
6:         end for
7:     end if
8: end for
9: return $\mathbf{G}$
```

As line 2 in Algorithm 1 indicates, whenever a new coding vector $\mathbf{R}$ is received, it is updated so that any previously decoded source packets are not considered again in the decoding process. If the updated row-vector $\mathbf{R}$ still contains non-zero entries, it is appended to the bottom of the decoding matrix $\mathbf{G}$ (lines 3-4). Lines 6-16 rearrange the rows of $\mathbf{G}$ in an effort to transform it into an upper triangular matrix. Lines 17-23 aim to transform $\mathbf{G}$ into row-echelon form by ensuring that each non-zero element on the main diagonal of $\mathbf{G}$ is the only non-zero element in that column. Finally, function `BackSubstitution` is called in line 25 to establish which source packets are decodable. The efficiency and accuracy of GE-PD are investigated in the following section.

IV. Results and Discussion

This section compares the proposed GE-PD with OFGE-PD, OFGE and GE in terms of computational cost and capability of progressively recovering source packets. The decoding algorithm that achieves the best accuracy but requires the least computational time is identified. It is then used to obtain simulation results, which are compared to theoretical predictions in order to validate the derived analytical expressions for systematic NC. The performance of systematic NC is then contrasted to that of straightforward NC and OU transmission, and the suitability of each scheme for progressive packet recovery is discussed.

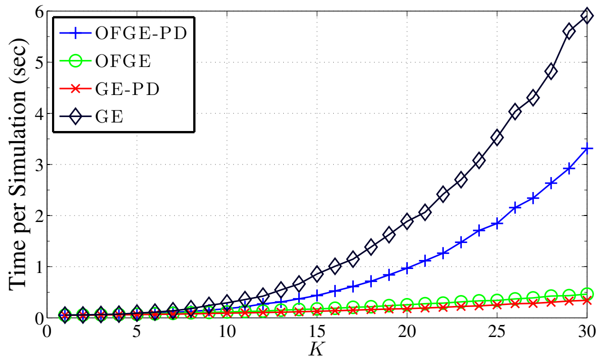


Fig. 1. Computational cost of the decoding schemes for different numbers of source packets $(K = 1, \ldots, 30)$.

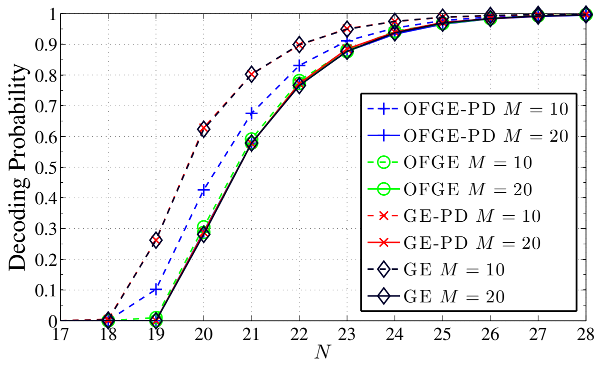


Fig. 2. Performance comparison of the decoding schemes for $K = 20$.

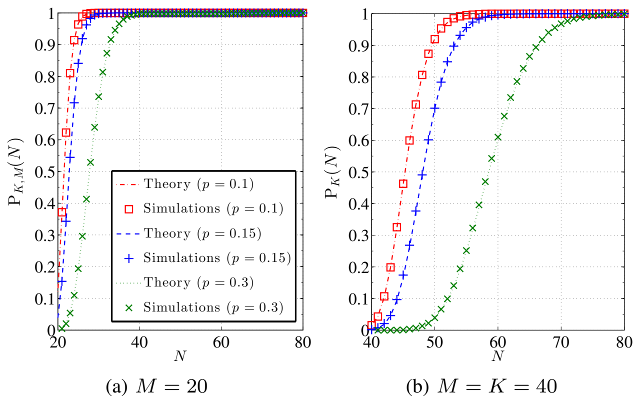


Fig. 3. Performance validation of systematic NC for $K = 40$, different values of p and (a) partial recovery $(M = 20)$ or (b) full recovery $(M = 40)$ of the source packets.

A. Assessment of GE-PD

Fig. 1 compares the *computational cost* of the considered decoding schemes. Recall that GE-PD and OFGE-PD are modified versions of GE and OFGE, respectively, which have been adapted to recover source packets from rank-deficient decoding matrices, as described in Section III. The computational cost has been expressed in terms of the time required for a decoder to recover the full sequence of K source packets when straightforward NC is applied and channel conditions are perfect, i.e. $p = 0$. The plotted results were obtained on a simulation platform equipped with an Intel Core i7-3770 processor and 8 GB of RAM. As expected [15], Fig. 1 shows that OFGE yields substantial computational savings over the conventional GE. However, the inclusion of progressive decoding capabilities in OFGE adds noticeable overhead to the decoding process. We observe that the computational cost of the resultant OFGE-PD increases rapidly for large values of K. On the other hand, GE-PD is not only more efficient than the original GE but also executes faster than OFGE.

Straightforward NC for $K = 20$ source packets and perfect channel conditions were also assumed for the performance assessment of the four decoding schemes. Fig. 2 depicts the probability of each scheme recovering at least half $(M = 10)$ or all $(M = 20)$ of the source packets when N packets have been transmitted. As we see, OFGE is not optimized for recovering a fraction of the source message in contrast to OFGE-PD, which requires a smaller number of transmitted packets to recover half of the source message but at a higher computational cost. A fact worthy of attention is that the decoding accuracy of GE is matched by that of GE-PD, which exhibits a computational cost as low as that of OFGE. For this reason, the proposed GE-PD was the decoding algorithm of choice for the simulation of the considered NC-based schemes.

B. Performance Validation of Systematic NC

In order to validate the derived analytical expressions for the decoding probability of systematic NC, a comparison between theoretical and simulation results was carried out. We considered a source message comprising $K = 40$ packets, which are encoded using a systematic NC and transmitted over a packet erasure channel with $p = \{0.1, 0.15, 0.3\}$.

Fig. 3a shows that expression (14) for $\mathrm{P}_{K,M}(N)$ accurately predicts the probability of decoding at least half of the source message $(M = 20)$. Similarly, expression (7) for $\mathrm{P}_K(N)$ matches the simulated results for decoding the entire source message $(M = 40)$, as reported in Fig. 3b. The excellent agreement between theory and simulation establishes the validity of the theoretical analysis. It also demonstrates that the proposed GE-PD is both efficient and accurate, considering that the number of decoded source packets matches the one predicted by the theoretical model.

C. Evaluation of Systematic NC for Progressive Decoding

Fig. 4 shows the probability that a receiver employing systematic NC recovers at least half $(M = K/2)$ or all $(M = K)$ of the source packets, when N packets have been transmitted. The performance of systematic NC is contrasted with that of OU transmission and straightforward NC, referred to here as SF NC for brevity. Two scenarios have been considered; Fig. 4a depicts the performance of the three transmission schemes when $K = 20$, while Fig. 4b presents plots for the case of $K = 40$. In both scenarios, the packet erasure probability has been set to $p = 0.1$.

We observe in Fig. 4a that OU transmission allows the recovery of at least half of the source message for a small value of N. However recovery of the whole source message requires a large number of transmitted packets. For example, for a target probability of $\hat{P} = 0.7$, a system using OU transmission can retrieve $M = 10$ source packets if just $\hat{N} = 11$ packets are transmitted. On the other hand, recovery of all $M = 20$ source packets requires the transmission of at least 39 packets. In other words, $\Delta N = 39 - 11 = 28$ packets need to be transmitted, on average, to allow recovery of the whole source message, when half of the message has already been retrieved. As we see in Fig. 4b, a larger value of K will markedly increase the value of ΔN.

By contrast, SF NC incurs a significant delay in recovering at least half of the source message but only a few extra transmitted packets are required to obtain the entire message. We observe in Fig. 4a that if $\hat{P} = 0.7$ then $\hat{N} = 24$ packets are needed to reconstruct half of the message, while the transmission of only $\Delta N = 1$ additional packet is sufficient for the decoding of the entire message.

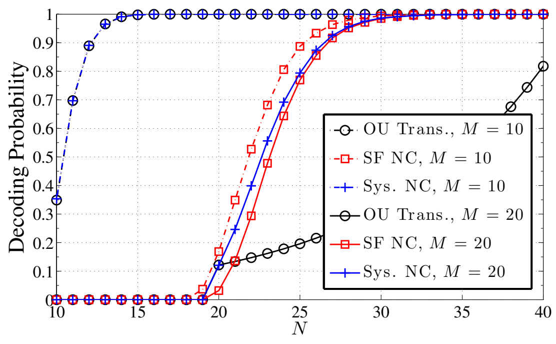
As is apparent from Fig. 4a and Fig. 4b, systematic NC combines the best performance characteristics of both OU transmission and SF NC. We observe that the value of $\hat{N}$ for recovering at least half of the source packets is as small as that of OU transmission, while the required number of transmitted packets for retrieving all of the source packets is smaller than or similar to that of SF FC. The latter observation confirms Proposition 2. Consequently, systematic NC is the most appropriate of the considered transmission schemes for progressive packet decoding, as it exhibits a high probability of either partially or fully decoding the source message.

V. Conclusions

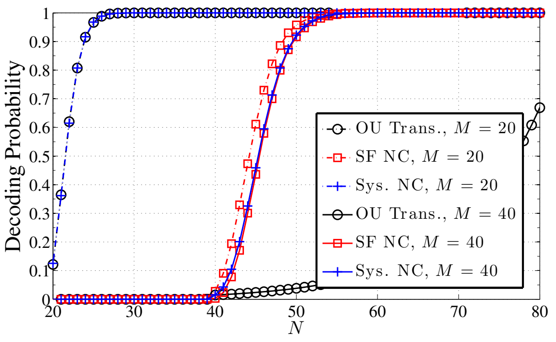
In this paper, we considered systematic random linear network coding, obtained theoretical expressions that accurately describe its decoding probability and proved that systematic network codes exhibit a higher probability of decoding the entirety of a source message than straightforward network coding. We also proposed Gaussian elimination for Progressive Decoding (GE-FD), which aims to recover source packets as soon as one or more transmitted packets are successfully delivered to a receiver. We demonstrated that GE-PD performs similarly to the optimal theoretical decoder in terms of decoding probability and also exhibits low computational cost. Furthermore, we established that the decoding delay characteristics of systematic network coding for both partial and full recovery of source messages are notably better than those of straightforward network coding.

Acknowledgment

This work was conducted as part of the R2D2 project, which is supported by EPSRC under Grant EP/L006251/1.



(a) $K = 20$



(b) $K = 40$

Fig. 4. Decoding probabilities as a function of N for $p = 0.1$ and (a) $K = 20$ or (b) $K = 40$.

References

[1] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[2] S. Zhang, S. C. Liew, and P. P. Lam, "Hot topic: Physical-layer network coding," in *Proc. MobiCom*, Los Angeles, USA, Sep. 2006.

[3] J. Heide, M. Pedersen, F. Fitzek, and T. Larsen, "Network coding for mobile devices - Systematic binary random rateless codes," in *Proc. IEEE ICC Workshops*, Dresden, Germany, Jun. 2009.

[4] D. Lucani, M. Médard, and M. Stojanovic, "Systematic network coding for time-division duplexing," in *Proc. IEEE ISIT*, Austin, USA, Jun. 2010.

[5] J. Barros, R. Costa, D. Munaretto, and J. Widmer, "Effective delay control in online network coding," in *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, Apr. 2009.

[6] R. Prior and A. Rodrigues, "Systematic network coding for packet loss concealment in broadcast distribution," in *Proc. ICOIN*, Kuala Lumpur, Malaysia, Jan. 2011.

[7] P. Saxena and M. Vázquez-Castro, "Network coding advantage over MDS codes for multimedia transmission via erasure satellite channels," in *Personal Satellite Services*. Springer International Publishing, 2013, vol. 123, pp. 199–210.

[8] O. Trullols-Cruces, J. Barcelo-Ordinas, and M. Fiore, "Exact decoding probability under random linear network coding," *IEEE Commun. Lett.*, vol. 15, no. 1, pp. 67–69, Jan. 2011.

[9] W. Koepf, *Hypergeometric Summation: An Algorithmic Approach to Summation and Special Function Identities*. Vieweg Verlag, 1998, p. 41.

[10] S. Roman, *The Umbral Calculus*. Academic Press, 1984, p. 29.

[11] S. Sesia, I. Toufik, and M. Baker, *LTE - The UMTS Long Term Evolution: From Theory to Practice*. John Wiley & Sons, 2011.

[12] A. L. Jones, I. Chatzigeorgiou, and A. Tassi, "Performance assessment of fountain-coded schemes for progressive packet recovery," in *Proc. CSNDSP*, Manchester, UK, Jul. 2014.

[13] J. Epperson, *An Introduction to Numerical Methods and Analysis*, 2nd ed. John Wiley & Sons, 2013, ch. 7, pp. 420–427.

[14] W. Niu, Z. Xiao, M. Huang, J. Yu, and J. Hu, "An algorithm with high decoding success probability based on LT codes," in *Proc. ISAPE*, Guangzhou, China, Nov. 2010.

[15] V. Bioglio, M. Grangetto, R. Gaeta, and M. Sereno, "On the fly Gaussian elimination for lt codes," *IEEE Commun. Lett.*, vol. 13, no. 12, pp. 953–955, Dec. 2009.

References

[1] A. L. Jones, I. Chatzigeorgiou, and A. Tassi, "Performance Assessment of Fountain-coded Schemes for Progressive Packet Recovery," in *Proc. 9th Int. Symp. on Commun. Syst., Networks and Digital Signal Process. (CSNDSP)*, Manchester, United Kingdom, Jul. 2014.

[2] M. Bogino, P. Cataldi, M. Grangetto, E. Magli, and G. Olmo, "Sliding-Window Digital Fountain Codes for Streaming of Multimedia Contents," in *IEEE Int. Symp. on Circuits and Syst. (ISCAS)*, May 2007, pp. 3467–3470.

[3] P. Cataldi, M. Grangetto, T. Tillo, E. Magli, and G. Olmo, "Sliding-Window Raptor Codes for Efficient Scalable Wireless Video Broadcasting With Unequal Loss Protection," *IEEE Trans. Image Process.*, vol. 19, no. 6, pp. 1491–1503, Jun. 2010.

[4] V. Bioglio, M. Grangetto, R. Gaeta, and M. Sereno, "On the fly Gaussian elimination for LT codes," *IEEE Commun. Lett.*, vol. 13, no. 12, pp. 953–955, Dec. 2009.

[5] L. L. Peterson and B. S. Davie, *Computer Networks: A Systems Approach*, 5th ed. Elsevier, 2012, ch. 2, pp. 102–106.

[6] A. F. Molisch, *Wireless Communications.* Chichester, 2011, ch. 4, pp. 47–51.

[7] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data," in *SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 4, Oct. 1998, pp. 56–67.

[8] M. Luby, "LT Codes," in *Proc. of the 43rd IEEE Symp. on Found. of Comput. Sci.*, Washington, DC, USA, Nov. 2002, pp. 271–280.

[9] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.

[10] M. Luby, M. Watson, T. Gasiba, T. Stockhammer, and W. Xu, "Raptor codes for reliable download delivery in wireless broadcast systems," in *3rd IEEE Consumer Commun. Networking Conf.*, vol. 1, Jan. 2006, pp. 192–197.

[11] *Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Content Delivery Protocols*, ETSI Techn. Spec., Rev. 1.3.1, Jun. 2009.

[12] A. L. Jones, I. Chatzigeorgiou, and A. Tassi, "Binary Systematic Network Coding for Progressive Packet Decoding," in *2015 IEEE Int. Conf. Commun. (ICC)*, London, United Kingdom, Jun. 2015, submitted for publication.

[13] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.

[14] D. J. C. MacKay, "Fountain codes," *IEE Proc. Commun.*, vol. 152, no. 6, pp. 1062–1068, Dec. 2005.

[15] J. Epperson, *An Introduction to Numerical Methods and Analysis*, 2nd ed. John Wiley & Sons, 2013, ch. 7, pp. 420–427.

[16] O. Trullols-Cruces, J. Barcelo-Ordinas, and M. Fiore, "Exact Decoding Probability Under Random Linear Network Coding," *IEEE Commun. Lett.*, vol. 15, no. 1, pp. 67–69, Jan. 2011.

[17] S. H. Friedberg, A. J. Insel, and L. E. Spence, *Linear Algebra*, 4th ed. Pearson, 2002, ch. 3, pp. 152–167.

[18] V. Bryant, *Aspects of Combinatorics: A Wide-Ranging Introduction.* Cambridge University Press, 1992, ch. 1, pp. 1–13.

[19] M. Spiegel, *Mathematical Handbook of Formulas and Tables*, ser. Schaum's Outline. McGraw-Hill, 1968, ch. 1, p. 3.

[20] S. Kim, K. Ko, and S. Chung, "Incremental Gaussian elimination decoding of raptor codes over BEC," *IEEE Commun. Lett.*, vol. 12, no. 4, pp. 307–309, Apr. 2008.

[21] W. Niu, Z. Xiao, M. Huang, J. Yu, and J. Hu, "An algorithm with high decoding success probability based on LT codes," in *9th Int. Symp. on Antennas Propagation and EM Theory (ISAPE)*, Nov. 2010, pp. 1047–1050.

[22] N. Rahnavard, B. Vellambi, and F. Fekri, "Rateless codes with unequal error protection property," *IEEE Trans. Inf. Theory*, vol. 53, no. 4, pp. 1521–1532, Apr. 2007.

[23] C. Studholme and I. Blake, "Windowed erasure codes," in *IEEE Int. Symp. on Inform. Theory*, Jul. 2006, pp. 509–513.

[24] D. Vukobratović, V. Stanković, D. Sejdinović, L. Stanković, and Z. Xiong, "Scalable Video Multicast Using Expanding Window Fountain Codes," *IEEE Trans. Multimedia*, vol. 11, no. 6, pp. 1094–1104, Oct. 2009.

[25] D. Vukobratović and V. Stanković, "Unequal Error Protection Random Linear Coding Strategies for Erasure Channels," *IEEE Trans. Commun.*, vol. 60, no. 5, pp. 1243–1252, May 2012.

[26] G. L. Mullen and D. Panario, *Handbook of Finite Fields*, ser. Discrete Mathematics and Its Applications. CRC Press, 2013, ch. 2, pp. 13–14.

[27] D. Vukobratović, C. Khirallah, V. Stanković, and J. Thompson, "Random Network Coding for Multimedia Delivery Services in LTE/LTE-Advanced," *IEEE Trans. Multimedia*, vol. 16, no. 1, pp. 277–282, Jan. 2014.

[28] Fraunhofer HHI, "JSVM Reference Software," http://bit.ly/V9lQ5M, 2014, [Online; accessed 10th Aug. 20014].