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

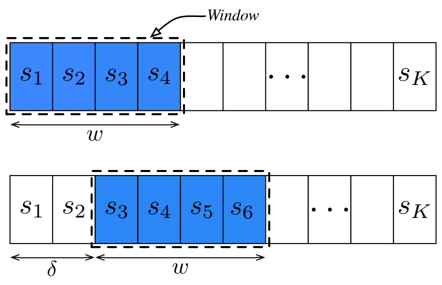
In this section, we describe the fountain-coded schemes under consideration. In all cases, a message comprising K source packets $s_1, s_2, \ldots, s_K$, is input to an encoder. The encoder generates N packets, $t_1, t_2, \ldots, t_N$, and transmits them over a broadcast erasure channel without feedback.

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

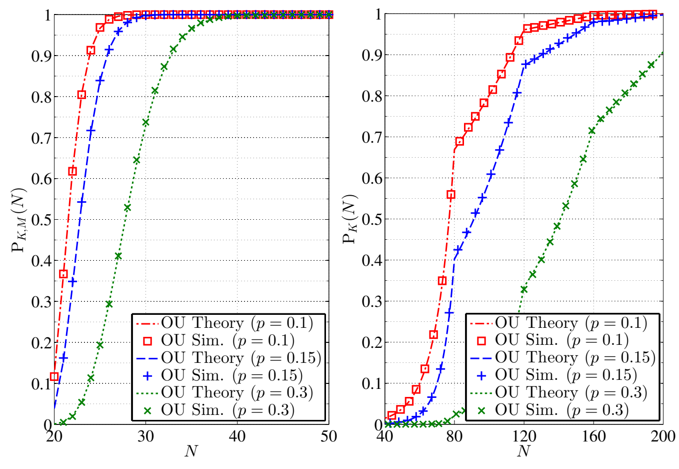
Having defined $\mathrm{P}_K(N)$ and $\mathrm{P}_{K,M}(N)$, we shall now obtain closed-form expressions for the case of Ordered Uncoded (OU) transmission, which will be used as a performance benchmark in our study. Note that the term *uncoded* transmission implies that the transmitted packets are not 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

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

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 $\mathrm{P}_K(\hat{N} + \Delta N) \geq \hat{P}$.

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 $\hat{P}$.

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

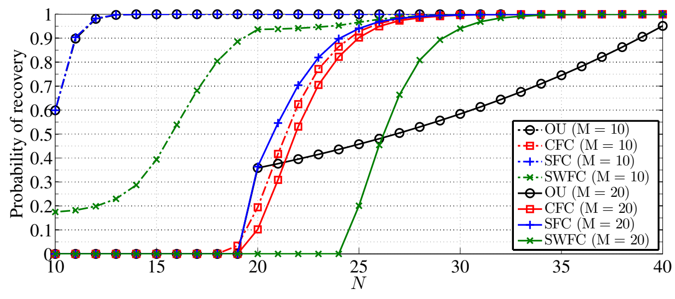
An example is given in Table I for OU transmission, $K = 40$, $M = K/2 = 20$ and $\hat{P} = 0.9$. The depicted values of $\hat{N}$ and ΔN generate probabilities $\mathrm{P}_{K,M}(\hat{N})$ and $\mathrm{P}_K(\hat{N} + \Delta N)$ that approach from above and are as close as possible to 0.9. They can both 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

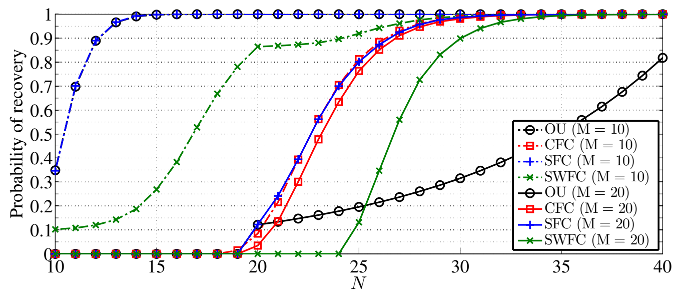
The responses of the reviewed schemes for $K = 20$ and $p = \{0.05, 0.1\}$, are shown in Fig. 3. It is apparent for this scenario that, with the aforementioned erasure channel, the SFC scheme is most appropriate. This is because SFC not only outperforms SWFC and CFC in terms of *progressive* packet recovery (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

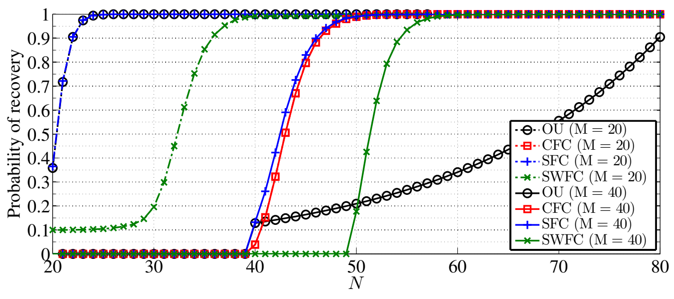


(a) $p = 0.05$

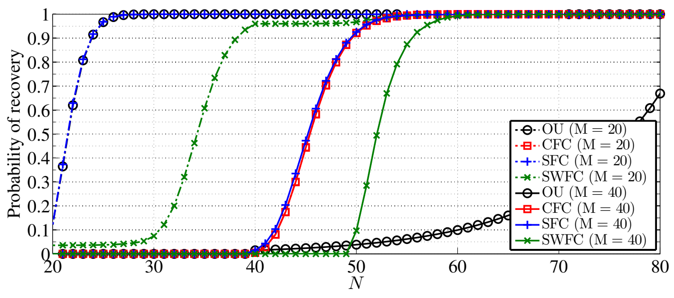


(b) $p = 0.1$

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.