2009 Fourth International Conference on Systems and Networks Communications

Sliding Window-Based Erasure Correction Using Biased Sampling

Tuomas Tirronen
Department of Communications and Networking
Helsinki University of Technology TKK, Finland
Email: tuomas.tirronen@tkk.fi

***Abstract*—We study a packet erasure correction method based on biased sampling in a sliding window. This kind of coding works especially well for streaming data, where the data have real-time requirements and expire after some time. The coding method resembles fountain coding but we allow biased sampling of the source blocks inside the window. We present an exact Markov model for analyzing the erasure correction performance and use Wallenius' noncentral hypergeometric distribution for calculating the erasure correction probability of a specified sampling pattern. The model is derived for independent packet erasures, and the sender is assumed to have an estimate on the erasure probability. We truncate the state space in order to lower the computational complexity by limiting the number of allowable erasures within the window, and study scenarios with low channel erasure probability and small window size. The optimal sampling weights are searched and the results suggest that a deterministic sampling pattern is optimal in the depicted scenario. A comparison to state-of-the-art Raptor codes for a similar setting reveals improved erasure correction performance with the proposed method.**

***Keywords*-performance analysis; erasure coding; real-time traffic; fountain coding**

I. Introduction

Erasure correction coding methods can be used to reduce the number of missing packets when transferring packet based data over networks which incur packet losses. A particularly efficient class of packet erasure correction codes are known as fountain codes, such as LT (Luby Transform) [1] and Raptor [2] codes. These codes do not rely on a feedback channel for acknowledging received data. This makes efficient multicasting possible, and situations like feedback implosion when, for example, downloading content from a server can be avoided. Overviews of fountain coding techniques can be found for example in [3] and [4].

In this paper, we consider a scenario where applications rely on using streaming techniques for content transfer, a popular setting on the Internet with many audio and video streaming services. We propose a systematic sliding window-based erasure correction method for protection against packet erasures in stream-type traffic. The source stream is divided into equisized blocks, which are to be transferred over a channel (network) with erasures. A sliding window defines the part of the stream the encoder is working on. The erasure correction method itself is based on sampling source blocks inside the window and generating linear combinations using the exclusive-or (XOR) operation. Our method is systematic in that each block is initially sent as-is. After each systematic packet, a repair packet is generated with probability P_r. Repair packets are generated by first sampling a degree distribution to pick a degree d, which is subsequently used to sample d blocks in the window for inclusion by a bitwise XOR in a repair packet.

Typically different kinds of decoding algorithms can be defined for these kinds of codes. For the sliding window method we consider iterative decoding: the already decoded blocks are subtracted from received repair packets possibly revealing novel decoded blocks. The principle of the decoding algorithm is the same as used in LT coding [1], the first universal fountain code. The encoding procedure also closely resembles the one used in LT encoding, with the exception of allowing biased sampling of the blocks. Thus, the presented erasure correction strategy has close ties to fountain coding. However, the optimal composition of the correction packets depends on the erasure channel parameters violating the nice channel independence-property typical to fountain codes. The presented coding method also has a code rate which depends on the repair packet sending probability P_r, i.e., the code is not strictly rateless; thus we cannot present the method as a pure fountain code. Nevertheless, the probability P_r is a continuous parameter and as a result any code rate between 0.5 and 1.0 can be achieved.

Our research problem is to determine the optimal way to generate repair packets. The performance metric we are considering is the residual packet loss probability. In previous work [5] we presented an exact Markov model to analyze the residual loss probability when the repair packet degree is first sampled and the blocks are then selected uniformly at random from the current window. This work presents an extension to the previous work by making the block sampling process biased, by allowing different sampling probabilities for each window position.

In particular, we calculate the probability of sampling a specific repair packet utilizing Wallenius' noncentral hypergeometric distribution [6]. We modify the previously presented model to accommodate biased sampling in order to evaluate the performance of the coding method analytically. Further, we validate the model by comparing the analytical results with simulations. As in [5], we need to restrict the maximum allowed number of packet drops in a window in order to make the state space small enough for viable

978-0-7695-3775-7/09 $26.00 © 2009 IEEE
DOI 10.1109/ICSNC.2009.93

computation. The magnitude of the resulting truncation error is studied by comparing the analysis to simulations with unrestricted number of packet drops.

The result of our previous study was that a fixed degree gives the optimal results when the constituents of a repair packet are sampled uniformly at random. This paper improves on that result and we will see that in addition to a fixed degree, a fixed deterministic pattern, corresponding to using weights $\{0, 1\}$ in the sampling distribution, gives optimal residual error performance. The erasure correction probability of such a pattern can be actually calculated without using the hypergeometric distribution resulting in fast calculation and making it possible to exhaustively search the whole state space for the optimal pattern. Nevertheless, the more complete model based on the Wallenius' distribution is a necessary intermediate step for ruling out other forms of sampling patterns and in achieving the final conclusion of the optimality of the deterministic patterns.

An important finding is that the form of the optimal pattern is essentially the same for different rate parameters for a given window size and erasure probability. It depends only on the used optimal degree. The type of the optimal pattern is such that few first and last positions of a window are always picked along with of more random pattern of some middle positions.

We will also present a comparison of the performance of the proposed scheme with that of Raptor coding, the current state-of-the-art fountain coding method. Discussion on the differences of the methods and scenarios where sliding window erasure correction performs better is presented.

The rest of this paper is organized as follows: Section II reviews the erasure coding method and analysis methods developed in [5]. In Section III, we show how to use Wallenius' noncentral hypergeometric distribution for modeling biased sampling of stream blocks from the sliding window. In addition, we present the needed modification for the analytical model. Numerical results using the analysis and simulations are presented and discussed in Section IV, with comparison to previous results and Raptor coding. In Section V, we recap our research on different strategies for real-time erasure coding and discuss the observations made. Finally we conclude in Section VI.

II. Review of Previous Work

In [5] we presented a performance analysis for calculating residual packet loss probability of sliding window erasure coding. The method is based on a Markov model, and can be used for optimizing the sent repair packets. In this section we give a review on the coding and the performance analysis methods. The idea of using a sliding window is inspired by [7].

A. Sliding Window Based Erasure Correction

The basic assumption is that the stream is transferred over an erasure channel, where packet losses are independent with loss probability p. The stream is divided into equisized blocks, which are initially sent intact. That is, we propose a systematic coding method, which has the benefit of allowing less capable recipients to access the passed original information without decoding. Redundancy is added by sending repair packets probabilistically after each original source block. We denote P_r as the repair packet sending probability. Then, the design code rate is $R = 1/(1 + P_r)$.

The coding method operates with a sliding window of fixed size w (in blocks). The sliding window is used to denote current and valid data the streaming sender wants to send. The window moves one step forward on the stream after every sent systematic packet, i.e., a new stream block is introduced into the first position of the window while one block is dropped from the end.

Repair packets are generated using degree distribution $\mathbf{\Omega}$, which is sampled for determining repair packet *degree*. A repair packet is composed of d sampled blocks in a window and summed using bitwise exclusive-or-operation.

The performance metric we consider is the residual packet loss probability Q_r. This is the probability for the last block inside the current window to remain unknown for the receiver before window movement and dropping out of the window.

In general, for different types of erasure and fountain codes, a number of different decoding algorithms can be defined. The coding process in these methods can be seen as generating linear combinations of the source blocks, and the receiver's job is to solve a system of linear equations. The most complete decoder would solve a full system of linear equations (e.g., with Gaussian elimination). However, iterative decoders, utilizing the concept of message passing, are computationally more efficient albeit with suboptimal equation solving capabilities.

The sliding window method we consider does not require solving a full system of linear equations. Instead, the decoder checks each of the arriving repair packets if it can be immediately used for decoding a novel block. Such a repair packet includes exactly one yet undecoded block and all of the other blocks have already been decoded by the receiver.

The decoder uses the information on which blocks are included in a repair packet[1] to check if the repair packet is immediately useful. If it is, all of the included blocks which are already decoded are subtracted (using bitwise XOR), which on average takes as many XOR-summations as the repair packet average degree, d_{avg}, is. If the repair packet does not have the correct form, it is discarded. It should be noted that the decoder does not need dedicated buffer space for decoding.

[1]Such information can be added as to header of the repair packets or calculated, e.g., by using the same random number generator at the sender and receiver(s).

B. Analytical Performance Evaluation

The coding and decoding process can be divided into three distinct steps: First window is moved by one step, then a systematic packet (the new packet in the window) is sent and finally, with probability P_r, a repair packet is generated and sent. For the last two steps there is probability p that the receiver does not receive the sent packet. These three steps make up a complete cycle, which is repeated over and over again. The state of the system over these cycles constitutes a Markov chain.

A state is described by the indices of the blocks in a window the receiver is missing. We denote by $\boldsymbol{i} = \{i_1, \dots, i_k\}$ the vector of the indices of k missing blocks where $i_1 < \dots < i_k$. As the number of possible states, 2^w, may be too large we limit the number of allowable errors in a window to 3. This keeps the state space viable for fast computation and is a realistic restriction when the window size w and channel loss probability p are low enough. Note that when no correction is used, the expected number of errors in a window is $w \cdot p$.

For each of the three steps we define a transition matrix corresponding to the possible state transition probabilities. We denote these matrices by $\boldsymbol{P}_1, \boldsymbol{P}_2$ and $\boldsymbol{P}_3$, respectively. Window movement, systematic sending and the potential repair packet generation constitute a full cycle, where the situation at the end of the cycle constitutes a Markov chain, the stationary distribution of which $\boldsymbol{\pi}$ can be solved from

$$\boldsymbol{\pi} = \boldsymbol{\pi} \cdot \boldsymbol{P}_1 \cdot \boldsymbol{P}_2 \cdot \boldsymbol{P}_3. \tag{1}$$

Our performance metric, the residual error probability Q_r, is the sum of the probabilities of those states where the last position of the window remains uncorrected, i.e.,

$$Q_r = \sum_{\boldsymbol{i}: i_k = w} \pi_{\boldsymbol{i}}. \tag{2}$$

We briefly list the composition of the three different state transition matrices, for details we refer the reader to [5]. The list includes only the strictly positive state transition probabilities.

The state transitions due to window movement are

$$\begin{cases} P_1\{\emptyset \to \emptyset\} = 1 \\ P_1\{(i_1, \dots, i_k) \to (i_1+1, \dots, i_k+1)\} = 1 \text{ if } i_k < w \\ P_1\{(i_1, \dots, i_k) \to (i_1+1, \dots, i_{k-1}+1)\} = 1 \text{ if } i_k = w, \end{cases} \tag{3}$$

where $\emptyset$ denotes the state with no errors.

Systematic sending occurs after every window movement step. The state transition probabilities are:

$$\begin{cases} P_2\{\boldsymbol{i} \to \boldsymbol{i}\} = 1 - p \\ P_2\{(i_1, \dots, i_k) \to (1, i_1, \dots, i_k)\} = p \quad \text{if } k < 3 \\ P_2\{\boldsymbol{i} \to \boldsymbol{i}\} = 1 \quad \text{if } k = 3, \end{cases} \tag{4}$$

where the last state transition probability is used to limit the number of possible errors and thus truncate the state space.

Lastly, the state transitions due to possible repair packet of degree d have the probabilities:

$$\begin{cases} P_3(d)\{\emptyset \to \emptyset\} = 1 \\ P_3(d)\{\boldsymbol{i} \to \boldsymbol{i} \backslash i_j\} = P_r(1-p)q(d, \boldsymbol{i}, i_j) \quad \text{if } j = 1, \dots, k \\ P_3(d)\{\boldsymbol{i} \to \boldsymbol{i}\} = 1 - \sum_j P_r q(d, \boldsymbol{i}, i_j)(1-p), \end{cases} \tag{5}$$

where $\boldsymbol{i} \backslash i_j$ denotes a state with index i_j deleted from $\boldsymbol{i}$. $q(d, \boldsymbol{i}, i_j)$ is the probability for a degree-d repair packet to immediately reveal a missing block i_j in state $\boldsymbol{i}$. For a degree-d repair packet with uniform sampling and k missing blocks in a window this is

$$q(d, \boldsymbol{i}, i_j) = \frac{\binom{w-k}{d-1}}{\binom{w}{d}}, \tag{6}$$

where $k = |\boldsymbol{i}|$.

To obtain the full $\boldsymbol{P}_3$ we sum $\boldsymbol{P}_3(d)$ over all possible degrees, i.e., the whole degree distribution:

$$\boldsymbol{P}_3 = \sum_{d=1}^{w} \Omega_d \cdot \boldsymbol{P}_3(d). \tag{7}$$

A single-degree distribution, where only one component of Ω_d is nonzero reduces the complexity as only one evaluation of matrix $\boldsymbol{P}_3$ is needed.

Next we will present the main contribution of this paper, namely allowing biased sampling in the repair packet generation and how to analyze the biased sampling strategy properly.

III. Modeling Biased Sampling

From this point on, we modify the scheme discussed above by introducing biased sampling in the window. That is, the degree d for repair packets is first chosen, then the window is sampled using a sampling distribution, or sampling weight pattern, $\boldsymbol{\omega}$, where component ω_i is the sampling probability (or weight) of block at index i in the window. The sampling is done without replacement, i.e., a block that is once sampled is not resampled for inclusion in the same repair packet again.

An example of the repair packet generation is depicted in Figure 1 for window size $w = 6$. New blocks are introduced from the left, i.e., the window on the figure moves one step left at the end of the cycle (after the possible repair packet sending). The repair packets are formed by first sampling d from degree distribution $\boldsymbol{\Omega}$ and thereafter picking d blocks using the sampling distribution $\boldsymbol{\omega}$.

Our goal is to calculate Q_r using the state transition probabilities for biased sampling, similarly as presented for uniform sampling in Section II-B. In Figure 1 Q_r refers to probability that the receiver does not have the block corresponding to index 6 decoded when the window

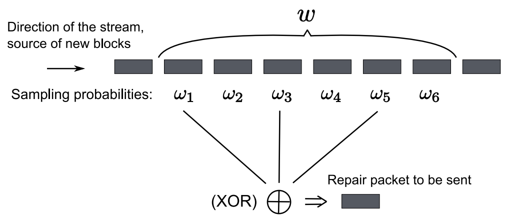
w
Direction of the stream, source of new blocks
Sampling probabilities: ω_1 ω_2 ω_3 ω_4 ω_5 ω_6
(XOR) $\bigoplus$ $\Rightarrow$
Repair packet to be sent

Figure 1. Example of repair packet generation for $w = 6$. The degree d is first sampled from the degree distribution (the result here is 3). After that d blocks are sampled using the sampling weights ω_i. As a result, in this figure, blocks corresponding to indices 1,3 and 5 are picked and summed together using bitwise XOR. The resulting repair packet is then sent to the channel.

moves. In order to calculate Q_r we need a way to calculate the probability of a generated correction packet to include specific blocks from the window.

A. Probability of Sampling a Specific Pattern

In general, hypergeometric distributions can be used for modeling picking colored balls at random from an urn without replacement. Weighed sampling (e.g., balls with different sizes) can be modeled using noncentral hypergeometric distributions. For the problem at hand, we use Wallenius' noncentral hypergeometric[2] distribution, where we agree beforehand on the number of samples d to be picked, i.e., we choose the repair packet degree in advance. Sampling of the blocks is done one after another, thus the total sampling probability mass remaining on the not-picked blocks has an effect on the next sample.

The original Wallenius distribution [6] is defined for two different categories of balls in an urn. We, however, use the multivariate version [9] which allows an arbitrary number of different weights. The probability mass function for a sample $\boldsymbol{x}$, where component x_i is the number of sampled balls from category i, is

$$P(\boldsymbol{x}) = \left(\prod_{i=1}^{c} \binom{m_i}{x_i}\right) \int_0^1 \prod_{i=1}^{c} \left(1 - t^{\omega_i/D(\boldsymbol{x})}\right)^{x_i} dt, \quad (8)$$

where c is the number of categories, m_i the initial number of balls in category i, and ω_i is the weight of balls in category i. Further, $D(\boldsymbol{x}) = \sum_{i=1}^{c} \omega_i(m_i - x_i)$ is the remaining total weight of the balls in the urn.

For our analytical approach, the number of categories equals the window size $c = w$, implying $m_i = 1$, i.e., a certain position in a window is either sampled once or not at all. We need to calculate the probability of a given sample $\boldsymbol{x}$, where $x_i = 1$ if the block at position i is sampled, and $x_i = 0$ if it is not, i.e., $\boldsymbol{x} \in \{0,1\}^w$. The ball weight ω_i corresponds to sampling probability of block at index i of the window. Equation (8) then simplifies to

$$P(\boldsymbol{x}) = \int_0^1 \prod_{i:x_i=1} \left(1 - t^{\omega_i/D(\boldsymbol{x})}\right) dt, \quad (9)$$

making the calculation of the binomial coefficients and some multiplications in (8) unnecessary. Form (9) is convenient when considering the calculation of the state transition probabilities, presented next.

B. Calculation of State Transition Probabilities

The performance can be computed analytically as presented in Section II-B. State transition probabilities (3), (4) and (5) are good as-is; we only need to alter the erasure correction probability $q(d, \boldsymbol{i}, i_j)$ to correspond to biased sampling.

In order to succeed in a state transition, the repair packet has to include the block which is going to be corrected while the other missing blocks defining the current state must not be included. Therefore, to calculate the probability of a certain state transition, we need to sum up (9) over all $\boldsymbol{x}$ including specific configuration of 1s and 0s. For example, if we are interested in the transition $\{i, j, k\} \rightarrow \{i, j\}$, we would need to sum up all $\boldsymbol{x}$ which include 1 at index k, 0 at i and j and the rest $d - 1$ ones in any of the remaining $w - 3$ places. The probability that a degree d repair packet can be used to resolve undecoded block at window position i_j is then

$$q(d, \boldsymbol{i}, i_j) = \sum_{x_{i_j}=1, x_{i_l}=0 \ \forall l \neq j} P(\boldsymbol{x}), \quad (10)$$

where $\boldsymbol{i}$ denotes the current state. Note that in the summation $l = 1, \cdots, k$ and $l \neq j$, where k is the total number of missing blocks in the state $\boldsymbol{i}$.

Especially if we have distinct sampling probabilities for each of the window positions, using (10) is computationally demanding. In the summation we need several evaluations of (9), which in turn includes $O(w^2)$ multiplications and an integral. For example, with $w = 12$ and thus 12 different categories (each block has its own probability to be sampled), we have $12 - 3 = 9$ free positions for 0s and 1s, and thus we would need up to 2^9 evaluations of (9). In this paper we will use maximum window size of $w = 12$, for which the calculations can be carried out in a reasonable time; using larger window sizes will result in impractically long computation times.

IV. Results

Using the presented analysis we can calculate exact Q_r fast for given set of parameters w, p, P_r, degree distribution $\boldsymbol{\Omega}$ and sampling weights $\boldsymbol{\omega}$. If the sampling is uniform, we know that the best performance is achieved using a single-degree distribution.

[2]This naming convention is suggested in [8].

Before we start to optimize the weight pattern itself, we study the truncation error incurred by restricting the number of errors in a window to three by comparing analytical and simulation results. This is required in order to show that the error is small enough to allow one use the analytical results in the optimization process. Subsequently we will find the optimal form of the weight pattern and give examples of the optimal patterns for different parameter sets. Finally we will compare the erasure correction performance between uniform and biased sampling and also compare the results with those obtained by using Raptor coding.

A. *Truncation Error with Biased Sampling*

Limiting the maximum number of errors in the window to three results in some truncation error in the analytical model. This error can be studied by comparing the analytical results with those obtained by simulations results.

Clearly the loss probability p and the window size w play an important role on the accuracy of the analysis. When the expected number of missing packets without correction, i.e., $w \cdot p$, grows, the truncation has more significant effect. The product $w \cdot p$ should be as low as possible, preferably of the order of 1.0 or less, to keep the effect of the truncation error small enough.

In Figure 2 we have plotted the residual error Q_r using (2) and compared simulation results with the same parameters and unlimited errors per window for $w = 8$ and various p. The degree distribution used is uniform, i.e., all eight degrees are equally probable. The sampling distribution is a geometric one, $\omega_i = 0.6^i / \sum_i 0.6^i$. Note that this distribution was chosen just to demonstrate the effect of the truncation error and we do not claim anything on its performance characteristics nor optimality. Figure 2 supports our guideline of keeping $w \cdot p$ of order of 1.0 or less, for $p = 0.20$, we have $w \cdot p = 1.6$ and the truncation results in a visible systematic error between analytical and simulation results. For lower p the error is not significant. Hence, our conclusion is that the analytical results can be used for optimization, as long as the product $w \cdot p$ is low enough.

B. *Biased Sampling - Optimal Weights*

With the described analytical tool we can now search for the type of optimal sampling weights. We start by using uniform sampling and see if we can improve on it by changing the sampling weights.

From [5] we see that for $w = 15, p = 0.1$ and $P_r = 0.2$ the optimal degree with uniform sampling is 10, with residual error $Q_r = 0.0459$. Now, we modify the analysis by introducing biased sampling and set for example the first weight so high that it is always picked and the rest are sampled uniformly, and calculate the performance. The resulting Q_r is 0.0452, which is lower than with uniform sampling. Similar scenarios with different parameters yield results indicating same kind of improvement. Relatively

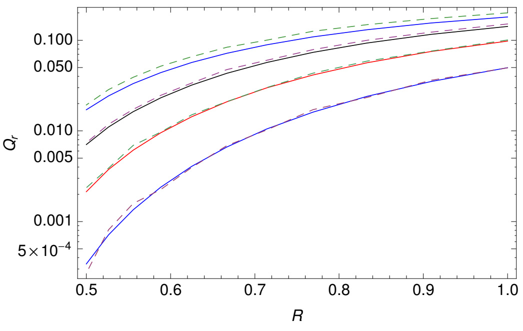

Figure 2. Effect of the truncation error for $w = 8$, $p = 0.05, 0.10, 0.15$ and 0.20, from bottom to top, respectively. Solid line is analytical and dotted the simulation result.

more significant difference can be seen when we modify more than just one sampling weight. Thus we conclude that biased sampling indeed is a viable option and can be used to improve the erasure correction performance.

The process of finding optimal biased weight pattern for any given set of parameters can be described as an iterative process. We can use as a starting point the results found using uniform sampling, i.e., we find the single degree optimal for the case. From there we can continue to modify the sampling distribution in order to find better weight pattern. With a better sampling distribution we can subsequently optimize the degree distribution. The analysis here yields the same results as conjectured in [5]: The optimal degree distribution is always single-degree type, i.e., a fixed d is used for sampling. This process can be iterated until optimal sampling weights and corresponding degree are found.

As an example of the process for finding optimal pattern, we have listed several evaluations of (2) in Table I for different weight patterns (not normalized) and degree distributions for $w = 6, p = 0.1$ and $P_r = 0.8$. We follow the list step by step in order to obtain the final, optimal, sampling pattern. We start with using degree three and uniform sampling. The performance is not very good, actually no erasure correction performance is noted at all. We continue by growing weight of position one and note an improvement in Q_r. After this, we raise the weight of every position in turn and find the pattern that gives the lowest residual probability (step III). Next we modify the degree distribution, and in steps IV and V find that using single degree of four yields better performance. After this we again find the best pattern where some of the positions have higher sampling weight than the rest (step VI). We see that growing the higher weights the performance increases ever more, also lowering the lower weights works out as well. Ultimately, the best performance is achieved by using the fixed pattern in step VIII. All further modifications of either $\boldsymbol{\omega}$ or $\boldsymbol{\Omega}$ result in worse performance.

Table I
EXAMPLE OF ITERATIVE PROCESS TO DETERMINE OPTIMAL FORM OF SAMPLING PATTERN FOR $w = 6, p = 0.1$ AND $P_r = 0.8$.

Step	Q_r	Sampling weights $\boldsymbol{\omega}$	Degree distribution $\boldsymbol{\Omega}$
I	0.0101	$\{1, 1, 1, 1, 1, 1\}$	$\{0, 0, 1.0, 0, 0, 0\}$
II	0.0083	$\{5, 1, 1, 1, 1, 1\}$	$\{0, 0, 1.0, 0, 0, 0\}$
III	0.0080	$\{5, 5, 1, 1, 5, 1\}$	$\{0, 0, 1.0, 0, 0, 0\}$
IV	0.0057	$\{5, 5, 1, 1, 5, 1\}$	$\{0, 0, 0.5, 0.5, 0, 0\}$
V	0.0042	$\{5, 5, 1, 1, 5, 1\}$	$\{0, 0, 0, 1.0, 0, 0\}$
VI	0.0040	$\{5, 1, 1, 5, 5, 5\}$	$\{0, 0, 0, 1.0, 0, 0\}$
VII	0.0033	$\{50, 1, 1, 5, 5, 5\}$	$\{0, 0, 0, 1.0, 0, 0\}$
VIII	0.0020	$\{1, 0, 0, 1, 1, 1\}$	$\{0, 0, 0, 1.0, 0, 0\}$

After performing such a process for any parameter set we find that optimal strategy is always to use a fixed, deterministic pattern. This means that $\boldsymbol{\omega} \in \{0, 1\}^w$, i.e., the correction packets are constructed of blocks taken from fixed positions in the window. Naturally, as the window moves every of the original blocks becomes under consideration on their turn. Thus the only probabilistic factor left is the repair packet sending probability P_r.

This observation on the form of the optimal pattern is based on extensive experimental testing. We do not have a rigorous proof on the optimal form of the weight pattern at this point, therefore the result should be regarded as a conjecture.

C. Deterministic Pattern

As we have found out, our results suggest that the best possible performance is achieved using a fixed degree and a deterministic pattern as the sampling distribution. For finding the weight pattern for which $\boldsymbol{\omega} \in \{0, 1\}^w$ we can use an exhaustive search. Moreover, when the sampling distribution is of such type, we do not need to use (9) in (10) when calculating the probability for decoding. Instead, the probability

$$q(d, \boldsymbol{i}, i_j) = \begin{cases} 1 \text{ iff } \omega_{i_j} = 1 \text{ and } \omega_l = 0 \; \forall \, l \in \boldsymbol{i} \backslash i_j \\ 0 \text{ otherwise } . \end{cases} \tag{11}$$

This results in fast calculation making an exhaustive search a viable method for finding the optimal pattern.

Table II presents the best pattern found and resulting residual error probability Q_r for $w = 12$ and $p = 0.10$. Blocks at window positions with weight 1 are always taken while weight 0 leaves the block out of the repair packet. We note that the optimal degree decreases with growing P_r.

An order of magnitude improvement to packet loss rate (i.e. p vs. Q_r) is achieved approximately with $P_r = 0.4$, which corresponds to code rate $R \approx 0.7$. The erasure correction performance against the code design rate is plotted in Figure 3 for $w = 12$ with $p = 0.05$ and 0.10. For $p = 0.10$ the plotted data are in Table II (column Q_r).

The performance difference between the best and worst possible deterministic pattern is also presented in Table II. The worst patterns is the same for all different parameter combinations, namely the pattern with one in the first

Table II
ANALYTICAL RESULTS USING DETERMINISTIC SAMPLING PATTERNS. CHANNEL LOSS PROBABILITY $p = 0.1$.

w	P_r	Q_r	pattern	d	diff. to worst
12	0.1	0.0625	{1,1,1,0,1,0,1,1,0,1,1,1}	9	0.0236
12	0.2	0.0392	{1,1,1,0,1,0,1,1,0,1,1,1}	9	0.0393
12	0.3	0.0230	{1,1,1,0,1,0,1,1,0,1,1,1}	9	0.0476
12	0.4	0.0119	{1,1,0,0,1,0,1,0,1,1,1,1}	8	0.0505
12	0.5	0.0058	{1,1,0,0,1,0,1,0,1,1,1,1}	8	0.0483
12	0.6	0.0025	{1,1,0,0,1,0,1,0,1,1,1,1}	8	0.0430
12	0.7	$9.5 \cdot 10^{-4}$	{1,1,0,0,1,0,0,1,0,1,1,1}	7	0.0358
12	0.8	$2.8 \cdot 10^{-4}$	{1,1,0,0,1,0,0,1,0,1,1,1}	7	0.0276
12	0.9	$5.9 \cdot 10^{-5}$	{1,1,0,0,1,0,0,1,0,1,1,1}	7	0.0189
12	1.0	$4.2 \cdot 10^{-6}$	{1,0,0,0,0,0,1,1,0,1,1,1}	6	0.0100

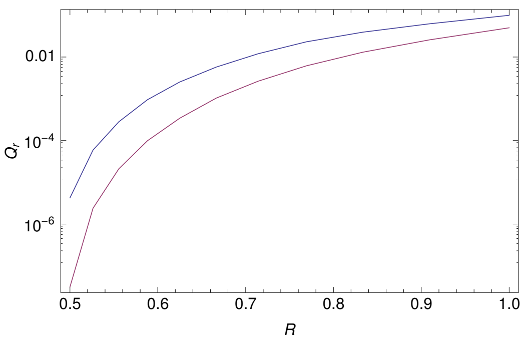

Figure 3. Performance of the sliding window coding for $w = 12$ and $p = 0.05$ (top) and 0.10 (bottom).

position and the rest zeros. Such a pattern is effectively the same thing as a repeat code, where every block is sent again with probability P_r.

The larger the window size w is, the better performance can be achieved. This is demonstrated in Figure 4, where we have plotted Q_r vs. code rate with $w = 6, 8, 10$ and 12 for a channel with erasure probability $p = 0.10$ using optimal patterns for each window size throughout. A larger window gives more possibilities for each of the missing blocks to be corrected during the process as each stream block is considered w times for inclusion in the repair packet. However, w is not a parameter that can be chosen at will, but the used application and the real time requirements dictate the size of the window.

The optimal patterns pick blocks both from the beginning and the end of the window. This is characteristic to all of the optimal patterns. This is understandable since if there were positions which are not chosen in the beginning or end of the window, then the same performance would be achieved by lowering the window size. However, larger window sizes have been observed to yield better performance, and thus the optimal sampling pattern always includes the first and last positions of the window. The exact formation of the ones in

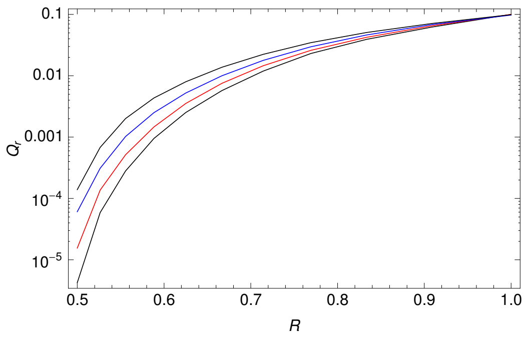

Figure 4. Comparison of different window sizes when $p = 0.1$. From top top bottom, $w = 6, 8, 10$ and 12.

Table III
THE EFFECT OF ERASURE PROBABILITY p ON THE OPTIMAL PATTERN.

w	p	P_r	Q_r	pattern	d
8	0.01	0.3	$8.6 \cdot 10^{-4}$	$\{1,1,1,1,1,1,1,1\}$	8
8	0.04	0.3	0.0065	$\{1,1,1,0,1,1,1,1\}$	7
8	0.07	0.3	0.0162	$\{1,1,1,0,1,1,1,1\}$	7
8	0.10	0.3	0.0297	$\{1,1,0,1,0,1,1,1\}$	6
12	0.01	0.3	$4.0 \cdot 10^{-4}$	$\{1,1,1,1,1,0,1,1,1,1,1,1\}$	11
12	0.05	0.3	0.0061	$\{1,1,1,0,1,0,1,1,0,1,1,1\}$	9
12	0.07	0.3	0.0115	$\{1,1,1,0,1,0,1,1,0,1,1,1\}$	9
12	0.10	0.3	0.0230	$\{1,1,1,0,1,0,1,1,0,1,1,1\}$	9

the middle of the window does not have a significant effect on the performance.

Table III lists optimal patterns when erasure probability p is varied for $w = 8$ and 12 and $P_r = 0.3$. We note that growing p has a similar effect as growing P_r: The optimal degree is lower with higher p. An important observation from these and also the results in Table II is that for a given window size and p the optimal pattern is dependent only in the optimal degree. For a specific degree used, the optimal pattern is always the same.

D. Comparison to Previous Results

In Table IV we have listed a comparison of the best achieved Q_r with biased sampling with the results in [5] using fixed degrees and uniform sampling for $w = 10$ and $p = 0.07$. By allowing biased sampling the residual error probability can be improved somewhat. The relative improvement is the better the lower the code rate is.

E. Comparison to Raptor Coding

As specified in [10] and [11], Raptor erasure correction coding [2] can be used for source sizes between 4 and 8192 source symbols with efficient decoding algorithm based on Gaussian elimination. We compare the residual error probability obtained by Raptor coding with those presented above for sliding window erasure correction for $w = 12$.

The methods differ inherently in some significant ways: Raptor coding operates on source blocks of size K source symbols. The encoder sends as many encoding symbols as required for decoding the source block or fails with some probability if not enough packets are received. The systematic version of Raptor coding sends original symbols as-is and afterwards repair symbols, where all encoding symbols are generated from intermediate symbols using LT coding [1]. The intermediate symbols are generated by a suitable precode, which is a fixed length block code. More extensive treatments of Raptor coding for multimedia transfer can be found in the literature, e.g. [12].

In other words, Raptor coding works more like traditional block codes when applied to our scenario of streaming traffic. Instead of looking for minimal number of repair packets in order to decode each original block with high probability, we search for optimal repair packets for making the residual error probability, Q_r, as low as possible with the probabilistic sending strategy specified in Section II.

In order to make a comparison of the methods possible, we allow the Raptor code to use the same amount of redundancy for the same amount of data as the sliding window coding. Thus, the Raptor encoder is allowed to generate a repair symbol for each original symbol it processes with probability P_r, i.e., the number of repair symbols for each set of K original symbols is binomially distributed with probability parameter P_r.

The decoder uses full Gaussian elimination to resolve the original symbols. In the case of a decoder error, i.e., when the number of received symbols is not enough for full decoding, the received systematic symbols are used.

Raptor coding can then be compared to the sliding window method by setting $K = w$. In Figure 5 we have depicted the simulated performance of both methods with $w = 12$ on an erasure channel with $p = 0.1$. As the repair symbols Raptor encoder actually is sending are always the same, different source symbols have different Q_r. Therefore, we have calculated the average Q_r for each set of w source symbols (i.e., each source block). The sliding window method gives better performance than Raptor coding when used as described above.

The first repair symbol the Raptor encoder sends when $K = 12$ consists of source symbols $1, 2, 3, 6$ and 7. This is the same thing as actually using sampling weights $\omega = \{1,1,1,0,0,1,1,0,0,0,0,0\}$. The erasure correcting performance of the Raptor code for $K = 12$ in the presented scenario could be increased by constructing first repair symbols corresponding to the patterns presented in Table II.

This comparison of the sliding window coding to Raptor code is not an exact one, as the methods differ in some fundamental ways. The Raptor encoder is artifically limited, and our method uses iterative, not full, decoding. However, the results give some indication that our method performs well when compared to the state-of-the-art methods.

Table IV
COMPARISON TO OLDER RESULTS (RESIDUAL PROBABILITIES)

w	p	P	old opt. prob (degree)	new opt. prob (degree)	opt. pattern
10	0.07	0.1	0.04226 (10)	0.04195 (9)	{1,1,1,1,0,1,1,1,1,1}
10	0.07	0.2	0.02599 (9)	0.02468 (8)	{1,1,1,0,1,1,0,1,1,1}
10	0.07	0.3	0.01569 (8)	0.01361 (8)	{1,1,1,0,1,1,0,1,1,1}
10	0.07	0.4	0.00934 (8)	0.00708 (7)	{1,1,0,0,1,0,1,1,1,1}
10	0.07	0.5	0.00543 (7)	0.00324 (7)	{1,1,0,0,1,0,1,1,1,1}

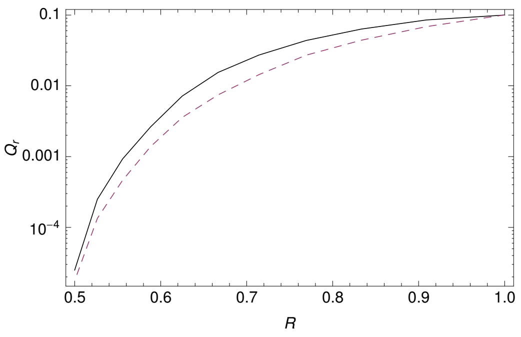

Figure 5. Comparison of biased sliding window coding (dashed) and Raptor coding (solid) for $w = 12$ and $p = 0.10$.

V. DISCUSSION

The results suggest that single-degree distributions actually give the optimal behavior for the proposed erasure coding method. A reasoning supporting this finding is as follows: Consider a certain window position with window size w, where the erasure probability is low enough, i.e. $w \cdot p$ is not much larger than one. This implies that most of the stream blocks in the window have already been received successfully, and only a couple of blocks need to be recovered using repair packet. For this to happen, there is no need to enter into long iterative decoding process of recovering missing blocks and packets of varied degrees as is the case for example in LT codes [1]. In LT decoding, the decoding process is kept going on by aiming at a new degree-1 block to be revealed on each decoding step. In the proposed method degree-1 blocks are abundant. Thus, the degree that gives the highest probability of all possible single degrees is chosen for the repair packet. Due to the sliding window-type coding, this degree stays the same as long as channel conditions are stationary. Therefore, it is reasonable to use only a fixed degree for a repair packet: there is no advantage in using a more complicated degree distribution.

A. Biased Sampling

As we can see from the results in Section IV, by allowing biased sampling the performance can be improved compared to uniform sampling and fixed degree erasure coding. The improvement is not a major one; nonetheless better performance is achieved by allowing different sampling weights. In particular, the optimal patterns are deterministic ones, where a fixed set of indices of the sliding window are always used for generating the repair packet. Thus ultimately there remains no possibility for probabilistic behavior, save the sending of repair packet. The degree remains fixed, and corresponds to the number of places which are always picked for the repair packet.

The optimal form of the deterministic pattern is such where first and last block in the window are always included. Typically, the larger the degree the longer are the continuous patterns of ones in the beginning and the end of the deterministic pattern. The pattern of middle positions which are always picked seem to be random, see e.g. II. There is no intuitive explanation for this. However, changing the exact positions of picked blocks in the middle of the optimal patterns does not make the results significantly worse, thus there are many good patterns which are close to the optimal one performance-wise.

VI. CONCLUSION AND FUTURE WORK

We have presented an erasure correction method based on sliding window. Our method for repair packet generation differs from previously published erasure correcting codes by allowing biased sampling without replacement of the original blocks. The proposed scheme works on erasure channels and is systematic, making it possible for a potential less capable receiver to utilize original data without decoding. The improvement on the residual error probability Q_r, compared to the worst case $Q_r = p$, the coding gives depends on the amount of redundancy used. This is controlled by the probability P_r with which redundant repair packets are sent after each systematic packet.

An exact analysis of the residual erasure probability with the proposed coding scheme is given, where, however, the maximum number of errors within the window is limited in order to lower computational complexity. A comparison of the analytical results with the simulated performance without this restriction reveals that the analysis is accurate for cases where the expected number of packet erasures, when no correction is used, is low.

Extensive studies on the form of optimal sampling weights suggest that the structure of the repair packets should always be the same: blocks in certain positions in the window are always picked and added together using bitwise XOR. This also means the repair packet degree is always the same. No performance benefit is found with degree distribution giving positive probability for multiple degrees. Also, the form of the optimal pattern stays the same for any given optimal degree as long as channel conditions and the window size are kept fixed. The few first and last positions in a window are the most important and should be included in repair packets in all cases.

The performance of the proposed method was further compared with Raptor coding for similar setting. The sliding window method, optimized for specific erasure probability and window size, gives better residual error performance than the Raptor coding.

The proposed method is easy to understand and apply to different kind of scenarios. It provides an interesting and efficient alternative to other well known erasure coding methods for providing robustness against packet losses for applications utilizing streaming traffic for content transfer.

Further research could be carried out to optimize the coding for possible correlated loss patterns and to study the effect of possible window size variation (e.g., due to jitter) during the sending process.

Acknowledgement

The work was financed by Graduate School in Electronics, Telecommunications and Automation, GETA and partly by ABI project funded by TEKES. In addition, the work was supported by a scholarship from the Nokia Foundation.

References

[1] M. Luby, "LT codes," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002, pp. 271–282.

[2] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[3] D. J. MacKay, "Fountain codes," *IEE Proceedings Communications*, vol. 152, no. 6, pp. 1062–1068, Dec. 2005.

[4] M. Mitzenmacher, "Digital fountains: a survey and look forward," in *Information Theory Workshop, 2004. IEEE*, Oct. 2004, pp. 271–276.

[5] T. Tirronen and J. Virtamo, "Performance analysis of sliding window based erasure correction for real-time traffic," in *Proceedings of Next Generation Internet Networks 2009, NGI'09*, 2009, to appear.

[6] K. Wallenius, "Biased sampling: The non-central hypergeometric probability distribution," Ph.D. dissertation, Stanford University, 1963.

[7] M. C. Bogino, P. Cataldi, M. Grangetto, E. Magli, and G. Olmo, "Sliding-window digital fountain codes for streaming of multimedia contents," in *IEEE International Symposium on Circuits and Systems*, May 2007, pp. 3467–3470.

[8] A. Fog, "Calculation methods for Wallenius' noncentral hypergeometric distribution," *Communications in Statistics - Simulation and Computation*, pp. 258–273, 2008.

[9] J. Chesson, "A non-central multivariate hypergeometric distribution arising from biased sampling with application to selective predation," *Journal of Applied Probability*, vol. 13, pp. 795–797, 1976.

[10] "3GPP TS 26.346 v.8.1.0 Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Service; Protocols and Codecs," 3GPP, Tech. Rep., Dec. 2008, available online at http://www.3gpp.org.

[11] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer, "RFC 5053: Raptor forward error correction scheme: Scheme for object delivery," IETF, Tech. Rep., Oct. 2007.

[12] M. Luby, M. Watson, T. Gasiba, T. Stockhammer, and W. Xen, "Raptor codes for realiable download delivery in wireless broadcast systems," in *Proceedings of IEEE Consumer Communications and Networking Conference, CCNC 2006*, Jan. 2006, pp. 192–197.