Publication 6

Tuomas Tirronen and Jorma Virtamo. 2009. Performance analysis of sliding window based erasure correction for real-time traffic. In: Proceedings of the 5th Conference on Next Generation Internet Networks (NGI 2009). Aveiro, Portugal. 1-3 July 2009, pages 1-8.

Performance Analysis of Sliding Window Based Erasure Correction for Real-Time Traffic

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

Jorma Virtamo
Department of Communications and Networking
Helsinki University of Technology TKK, Finland
Email: jorma.virtamo@tkk.fi

***Abstract*—We propose a fountain-coding-like erasure correction method for streaming traffic with real-time requirements. A sliding window defines the range of non-expired data. Each new block entering the window is once sent as such, followed by probabilistically sending a repair packet, formed as a random combination of the blocks in the current window using a degree distribution as in LT-coding. The decoding of repair packets is based on an iterative algorithm. The performance of the method with a given channel loss probability is analyzed using an exact Markov chain model. The state space, however, has to be truncated for computational tractability. The truncation error is verified to be small enough by simulations. By using the analytical model the optimal degree distribution is found to be of single-degree type. Both analytical and simulation results on the performance with the optimal repair packet degree are presented and the effectiveness of the method demonstrated.**

Keywords: performance analysis, erasure coding, real-time traffic, simulation

I. Introduction

Media streaming is a popular application on today's networks, especially on the Internet. Fast consumer broadband access, ever increasing computing capacity and the multitude of different devices, even handheld terminals using wireless access, have made streaming a common and viable method for transferring media content. Video streaming makes it possible to simultaneously download and play back video clips instead of waiting for a full download to complete. Similarly, different kinds of voice-over-IP (VoIP) services and Internet radio have emerged relying on streaming audio data.

Streaming media over networks poses several challenges relating to the quality of the content delivery and places some requirements on the underlying network conditions, like enough bandwidth, bounded end-to-end delay, and low enough jitter and packet losses [7]. On the Internet, with best-effort service, this kind of quality of service (QoS) guarantees cannot be usually assured.

Coping with packet losses with real-time requirements is in the focus of this paper. In general, several different methods can be employed to increase the robustness of streaming against packet losses. We propose and analyze a specific, fountain-code-type erasure coding method for this purpose and demonstrate its effectiveness in terms of the code rate vs. residual packet loss performance.

Fountain codes were originally introduced for sending a given message over a lossy channel without any real time requirements [2]. In fountain coding the transmitter acts as a digital fountain able to produce a virtually infinite number of different encoded packets. Each of the encoded packets is formed of the blocks of the original message by a stochastic algorithm in identical way, independent of the others. Thus none of them counts more than another, making the code channel independent; only the number of received packets matters.

In applying the fountain coding principle to streaming media we utilize a sliding window, similarly as in [1]. The sliding window defines the part of the stream the encoder is working on and its size is directly determined by the real-time requirements of the used application.

Our proposed method is systematic in that each new block of the stream entering the window is sent once as such. This is a natural and good idea in the case of streaming media. Following the systematic packet, a repair packet is sent with a given probability P_r, with the intention to fill possible gaps in the receivers window of successfully received and decoded blocks. P_r is a planning parameter that controls the trade-off between the code rate and the residual probability of missing blocks.

In the chosen approach each repair packet is generated in similar fashion as in LT-codes [5], the first universal fountain codes, by forming a linear combination of a random sample of original blocks in the current window. The number of blocks in each sample, so-called degree of the repair packet, is drawn from a degree distribution. One of the main objectives of this paper is to find the optimal degree distribution assuming that the channel loss probability p is known. For the presented method to work, all of the blocks are required to be of the same size. Thus, in practice, the method would suit better for applications where equal sized blocks are used. This means many video-streaming applications are excluded without extending the model to work for varying sized packets.

We analyze the performance of proposed method analytically by modeling the evolution of the pattern of missing blocks in the receiver's window as a Markov chain. The state of the system at the end of each complete cycle consisting

*The work of Tuomas Tirronen was financed by Graduate School in Electronics, Telecommunications and Automation, GETA and partly by ABI project funded by TEKES. In addition, part of the work was done with the help of scholarship from Nokia Foundation.

978-1-4244-4245-4/09/$25.00 (c)2009 IEEE

of moving the window one step forward and sending the new packet, possibly followed by sending a repair packet, constitutes a time-homogeneous Markov chain. We solve the stationary state probabilities of this chain and deduce from these the residual probability of a block still to be missing just before its data expire. The method of analysis is exact but we need to limit the number of missing packets in a window in order to avoid state space explosion. The analytical results allow us to evaluate the performance of the system quickly and facilitate optimizing the degree distribution for the repair packets.

In a previous work [6], we studied a method where each block of the window was included in the repair packet independently with a position dependent probability (i.e., the degree was not directly controlled), and presented an analysis to optimize these probabilities. The analysis necessitated making some independence assumptions and resulted, in the light of the present results, in a suboptimal method.

We examine the error in the analytical results due to the truncation of the state space by comparing them with simulations and find them accurate enough for studying the optimal degree distribution. An important finding is that a fixed degree (as opposed to a distribution) that depends on the parameters of the problem seems to realize the optimum. The performance of the proposed method is then extensively studied by simulations by using optimized repair packet degrees.

We also present the Gilbert-Elliot model for simulating correlated losses and give simulation results with bursty packet losses using the degree distributions optimized for independent losses. We find that an extended model with taking different loss patterns into account would result in better performance.

The rest of this paper is organized as follows: In Sect. II we describe the problem setting Sect. III we present a Markovian analysis of the system and derive the probability for blocks to remain undecoded. The performance is evaluated using simulations, and the results along with a brief discussion are given in Sect. IV. Finally we conclude in Sect. V.

II. Sliding Window Based Erasure Coding

A. Problem Setup

We assume an application sending streaming data with real-time requirements to one or possibly several users. The streaming data are assumed to be divided into equisized *blocks*. An encoding symbol is either an original block as-is or a linear combination of several blocks, called a *repair packet*. We do not have any restrictions on the symbol (i.e., block and packet) size.

We model a lossy network as a packet erasure channel with erasure probability p. Thus a packet is assumed to be received correctly with probability $1-p$ and dropped with probability p. The channel is assumed to be unidirectional with no explicit feedback channel available. However, we assume that there exists some mechanism for propagating the estimated channel erasure probability p to the sender. Knowing the erasure probability makes it possible for the encoder to make optimal decisions on the encoding strategy.

B. Sliding Window

We employ a sliding window to define the part of the source data the encoder is working on. The window size w is dictated by the real-time requirements of the used application. In this work we are interested in window sizes of tens. For example, in typical VoIP scenario voice data could be packed into blocks each including 10 or 20 ms of voice. For adequate quality the maximum allowed end-to-end delay is typically in the range of $150-200$ ms; thus the number of outstanding packets could be around ten, when different delay sources are taken into account.[1]

The sliding window determines the section of the stream that is currently valid and useful for the receiver. As the window moves one step, a new block is introduced to the beginning of the window (location 1) and at the same time an old block expires and is dropped from the other end (location w). The original blocks in different window locations are denoted s_i, $i=1,\ldots,w$. The code is systematic meaning that block s_1 is sent once as such. This is followed by sending a repair packet probabilistically as described in the next subsection. The process is repeated after each window movement. Ultimately, the probability that block w remains undecoded determines the residual error probability, the performance metric we are interested in. This is studied in detail in Sect. III.

C. Repair Packet Generation

The repair packets, denoted here by r, are sent probabilistically with probability P_r after each systematic packet. P_r defines the design rate of the code,

$$R = \frac{1}{1+P_r} \ . \tag{1}$$

Thus the range possible of design rates with the proposed method is $[0.5, 1.0]$.

The repair packets are linear combinations of original stream blocks s_i over the binary field, meaning that the summation is performed by a bitwise XOR-operation ($\oplus$),

$$r = \bigoplus_{j=1}^{d} s_{i_j} \ , \tag{2}$$

where d, the number of blocks in the combination, is called the *packet degree*. It is drawn from a *degree distribution*, with Ω_d denoting the probability that degree d is chosen. Then, d blocks from the original window are sampled uniformly at random, i.e., the i_j are a random sample of size d of the set $(1,\ldots,w)$. The average degree of all sent packets is

$$d_{\text{avg}} = \Big(1 + P_r \cdot \sum_{i=1}^{w} i\Omega_i\Big)/(1+P_r). \tag{3}$$

[1]The delay budget is consumed by different sources of delay, e.g., processing, propagation and queueing. We do not consider these explicitly but assume the time a packet is allowed to stay in the play-out buffer is given and is constant.

D. Decoding

The decoding done in the same way as in LT-decoding [5], i.e., using an iterative belief propagation algorithm. The decoder needs description of the repair packet composition in order to perform the decoding. This can be accomplished for example by using a bitmask in the header of the packets for indicating the positions of sampled blocks, or by using a pseudo-random number generator with common seed on both ends [5].

When a repair packet arrives at the decoder, the decoder can immediately reveal a new block if the packet is linear combination of $d-1$ already decoded blocks and exactly one yet unknown block. The revealed block then is

$$s = r \oplus \left(\bigoplus_{j=1}^{d-1} s_{i_j} \right) , \tag{4}$$

where the indices i_j correspond to the already decoded blocks in the repair packet. The receiver can buffer those repair packets that are not immediately useful and subtract a received systematic packet from those buffered repair packets including it, possibly revealing novel blocks. The buffer is iteratively processed as long as it is possible to decode new blocks. The lower the mean degree of packets (3), the lower is the total time complexity of the decoding algorithm, as the iterative decoding needs to go through fewer steps, when decoding the repair packets.

An implementation of the decoding can be done at different levels of completeness and complexity with regard to how the repair packets that are not immediately useful are handled. In the sequel we will refer to three different variants, which in the order of increasing time and space complexity are:

I. The simplest method only considers repair packets which are immediately decodable. Repair packets are not buffered. The analysis in Sect. III is restricted to the use of this method.
II. A more complete receiver stores those repair packets which are not immediately useful in a buffer. The arriving systematic packets are subtracted from buffered repair packets possibly revealing novel blocks. A buffered packet is discarded once a block included in it expires and moves out of the window.
III. The most complete strategy does not discard any repair packets but allows decoding using out-of-window blocks.

E. Burst Erasure Model

Packet losses are often correlated in real-life scenarios. For example, if the losses occur due to network congestion, there are often multiple drops in a row or a period when drops are much more likely than under normal conditions. For wireless and wired links the bit errors are often modeled using Gilbert-Elliott (GE) model [3], [4].

In the GE model the channel is either in bad or good state, both of which have their own packet loss probability. We use the model with independent losses for each state, with packet erasure probabilities p_a and p_b in good and bad state, respectively called "OK" and "Burst" state. The changes of channel state are independent of previous states, thus the model constitutes a two-state discrete time Markov chain, depicted in Figure 1. The state transition matrix is

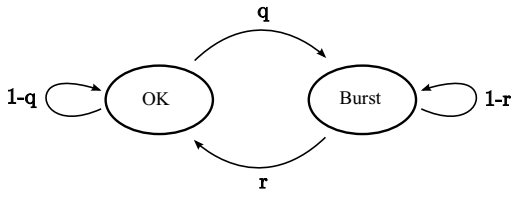

Figure 1. Gilbert-Elliott model. Both of the states have their own loss probabilities.

$$\boldsymbol{P} = \begin{pmatrix} 1-q & q \\ r & 1-r \end{pmatrix} \tag{5}$$

and the corresponding stationary state probabilities $\boldsymbol{\pi} = (\pi_{\text{ok}}, \pi_{\text{burst}})$ are

$$\begin{cases} \pi_{\text{ok}} = \frac{r}{q+r} \\ \pi_{\text{burst}} = \frac{q}{q+r}. \end{cases} \tag{6}$$

The average packet loss probability can be then calculated as

$$p_{\text{avg}} = p_a \cdot \pi_{\text{ok}} + p_b \cdot \pi_{\text{burst}}. \tag{7}$$

In Section IV we present simulation results using degree distributions obtained with the optimization method developed in the next Section. We concentrate on independent packet losses, however, for completeness we also give results and comparison using the GE model while using the optimal degree distributions for the independent loss model.

III. Probability of an Undecoded Block

Ultimately we are interested in the probability Q_r of an arbitrary packet remaining undecoded, i.e., the decoder error probability. This quantity works as a performance metric, similar to the residual BER for lower layer FEC or ARQ techniques. In this section we derive Q_r analytically under a two simplifying assumptions. First, we assume that only simple decoding, method I, is used, i.e., a repair packet that is not immediately useful for releasing a new decoded block is discarded. Secondly, we need to truncate the state space in order to get a tractable system. The idea with the analysis is to get results that can be computed relatively quickly, in comparison to simulations, and thus can be easily used for optimizing the degree distribution of the repair packets. The derivation is based on the analysis of the state of the system as a Markov chain.

A. State of the System and Markov Chain

A full description of the state of the system is given by the Bernoulli variables $x_i,\ i = 1, \ldots, w$, telling whether the block in the ith position of the window has been successfully decoded, $x_i = 1$, or not, $x_i = 0$. The whole state is thus

represented by the vector $\mathbf{x} = (x_1, \ldots, x_w)$. For our analysis, however, it is more convenient to use an equivalent notation where a state is identified by just listing the indices of missing blocks, i.e. the locations of zeroes in $\mathbf{x}$. The state vector is thus $\mathbf{i} = (i_1, \ldots, i_k)$, where k is the total number of missing blocks in the window and $i_1 < \cdots < i_k$. A state with no missing blocks is denoted by $\emptyset$.

This alternative state description as an index vector $\mathbf{i}$ is particularly handy when the number of missing blocks k is small. In fact, in order to avoid state space explosion (the total number of states is 2^w) we need to limit our analysis to states where the number of missing blocks is just 2 or 3. For example, the total number of states for $w = 15$ would be 32768. By limiting the number of missing blocks to 3 we achieve a reduction to 576 states and a further limitation to 2 missing blocks would result in 121 different states. The state space truncation introduces some error into the analysis but for small channel loss probabilities the error is not significant. The effect of the truncation will be considered in more detail in Sect. IV by comparing the results with those obtained by simulations without any artificial restrictions.

The state of the system evolves in the course of time. We consider the evolution over a full cycle consisting of three parts. First, as time elapses the block in position w expires and correspondingly a new position opens for a new packet to arrive at the beginning of the window. This step is called window movement. Second, a new packet of the stream is sent as such (systematic sending) and if successfully received occupies position 1 in the window; otherwise the packet in position 1 will be missing. Finally, a repair packet is sent with probability P_r and may release a new decoded packet somewhere within the window. To each of these steps there corresponds a state transition probability matrix, respectively denoted by $\mathbf{P}_1, \mathbf{P}_2$ and $\mathbf{P}_3$. The full transition probability matrix over the whole cycle is then $\mathbf{P} = \mathbf{P}_1\mathbf{P}_2\mathbf{P}_3$. The stationary distribution $\boldsymbol{\pi}$ of the state probabilities at the observation point at the end of each cycle can now be solved from

$$\boldsymbol{\pi} = \boldsymbol{\pi} \cdot \mathbf{P} = \boldsymbol{\pi} \cdot \mathbf{P}_1\mathbf{P}_2\mathbf{P}_3 \ , \tag{8}$$

along with the normalization condition $\boldsymbol{\pi} \cdot \mathbf{e}^T = \mathbf{1}$, where $\mathbf{e}$ is the vector of all ones. The residual error probability Q_r is the probability that at the end of the cycle, just before the window movement and expiration of the data at position w, the block in question is still undecoded. After solving the stationary distribution $\boldsymbol{\pi}$, the error probability Q_r is obtained by summing up the probabilities of the states in which block w is missing,

$$Q_r = \sum_{\mathbf{i}: i_k = w} \pi_{\mathbf{i}} \ . \tag{9}$$

Next we will construct matrices $\mathbf{P}_1, \mathbf{P}_2$ and $\mathbf{P}_3$. Only the non-zero elements of these matrices are considered.

B. Transitions Due to Window Movement

Window movement transfers the position of all blocks one step forward deterministically. The number k of missing blocks is reduced by one if block w was missing before the movement (as the position drops out of the window); otherwise the number stays unchanged. More specifically for $k = 0$ (no missing blocks) there is only the self-transition

$$P_1\{\emptyset \to \emptyset\} = 1 \ , \tag{10}$$

and for $k \geq 1$ we have

$$\begin{cases} P_1\{(i_1, \ldots, i_k) \to (i_1 + 1, \ldots, i_k + 1)\} = 1 & \text{if } i_k < w \\ P_1\{(i_1, \ldots, i_k) \to (i_1 + 1, \ldots, i_{k-1} + 1)\} = 1 & \text{if } i_k = w \end{cases} . \tag{11}$$

C. Transitions Due to Systematic Sending

The systematic packets are sent after window movement. The transitions depend on the channel erasure probability p. With probability p a missing block is introduced at position 1 and with probability $1 - p$ the state remains unchanged,

$$\begin{cases} P_2\{\mathbf{i} \to \mathbf{i}\} = 1 - p \\ P_2\{(i_1, \ldots, i_k) \to (1, i_1, \ldots, i_k)\} = p \end{cases} . \tag{12}$$

When the state space is artificially truncated so that no more than K missing packets are allowed, we enforce a state with $k = K$ to remain the same irrespective of whether the sent systematic packet is lost or not,

$$P_2\{\mathbf{i} \to \mathbf{i}\} = 1 \quad \text{for } k = K \ . \tag{13}$$

Obviously, this is a source of some error in the analysis.

D. Transitions Due to Repair Packets

Next we address the probability for a repair packet to immediately release some of the missing blocks. The transition probabilities depend on the degree d of the repair packet. We first consider the transition probabilities for a given d and denote the corresponding transition probability matrix by $\mathbf{P}_3(d)$. The actual transition matrix is then obtained by deconditioning.

When there are k missing blocks altogether in the window, then a repair packet of degree d is able to reveal a specific missing block with the probability

$$q(d, k) = \frac{\binom{w-k}{d-1}}{\binom{w}{d}} \ , \tag{14}$$

since a successful repair packets must include the given block as the only novel block (no choice) while all the other $d - 1$ blocks in the repair packet are chosen from the $w - k$ known ones. The denominator gives the total number of repair packets of degree d.

Now we can construct the matrix $\mathbf{P}_3(d)$. For $k = 0$ we again have

$$P_3(d)\{\emptyset \to \emptyset\} = 1 \ , \tag{15}$$

as there are no missing packets to be corrected. To study the case $k \geq 1$ note that a given missing block i_j in a window is repaired if and only if a repair packet is generated (probability P_r), it is not dropped by the channel (probability $1 - p$), and

Table I
DEGREE DISTRIBUTIONS USED IN STUDYING THE TRUNCATION ERROR FOR $w = 15$.

Name	Distribution
Uniform	$\Omega_i = 1/15 \approx 0.067 \quad \forall i$
All-1	$\Omega_1 = 1$ and $\Omega_i = 0$ for $i = 2 \dots w$
Hi-low	$\Omega_1 = 0, \Omega_2 = 0.4, \Omega_{15} = 0.4$ and $\Omega_i = 1/12 \approx 0.0167$ for $i = 3 \dots 14$

it is capable or revealing the missing block, the probability of which is given by (14). Thus we have

$$P_3(d)\{\mathbf{i} \to \mathbf{i}\backslash i_j\} = P_r(1-p)q(d,k) \quad \text{for } j = 1, \dots, k \ , \tag{16}$$

where $\mathbf{i}\backslash i_j$ denotes a state with index i_j deleted from $\mathbf{i}$, and

$$P_3(d)\{\mathbf{i} \to \mathbf{i}\} = 1 - k\, P_r(1-p)q(d,k) \ . \tag{17}$$

Now, using the degree distribution of the repair packets the state transition matrix $\mathbf{P}_3$ is obtained from $\mathbf{P}_3(d)$ as

$$\mathbf{P}_3 = \sum_{d=1}^{w} \Omega_d \cdot \mathbf{P}_3(d) \ . \tag{18}$$

It is evident that the computation becomes faster when the degree distribution has only few non-zero components, and a single-degree distribution needs only one evaluation of $\mathbf{P}_3(d)$.

IV. NUMERICAL RESULTS

In this section we first address the error in the analytical results in Sect. III due to the state space truncation. Having verified a sufficient accuracy we use the analytical results to study the form of the optimal degree distribution. Finally, we make simulations with the optimized distributions using the three different decoding variants presented in Sect. II-D. The erasure correcting performance is evaluated by studying the code rate vs. residual packet loss probability.

We consider scenarios with small window sizes and not too high loss probabilities. For example when $w = 15$ and $p = 0.15$ the expected number of missing blocks in a window is 2.25, without taking into account the corrections. These values are in line with the restriction of the maximum number of missing blocks to three in the analysis and represent typical parameters studied in this work.

A. Truncation Error

In order to justify the use of the analytical results of Sect. III for the optimization of the degree distribution, we study the truncation error by comparing the results with simulations without any restriction on the number of missing blocks in the window. For an unbiased comparison, the simple decoding method I is employed in both cases as it is an inherent assumption in the analytical model. The comparisons are made with the degree distributions presented in Table I. Figure 2 shows that the analytical results follow the simulations closely. The residual error probability given by the analysis is slightly lower than the simulated one, as the truncation artificially limits the number of missing blocks.

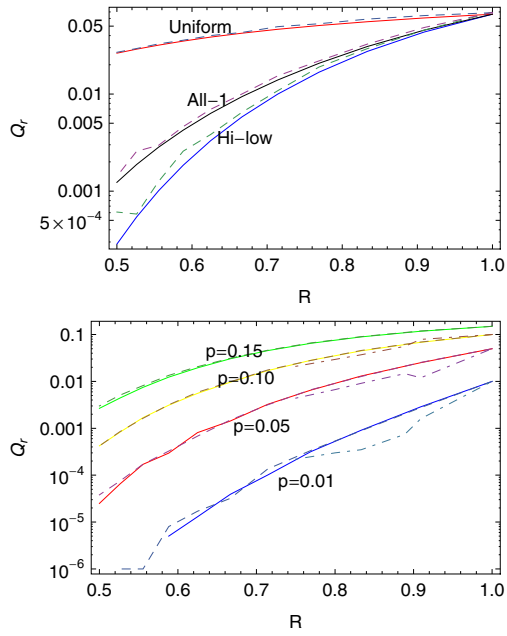

Figure 2. (Top) Comparison of analytical and simulation results for the three degree distributions in Table I for $w = 15$. (Below) Comparison of the performance using degrees optimized by the analysis and optimal degrees found using exhaustive search from simulations for $w = 15$.

B. Type of Optimal Degree Distribution

The analytical result (9) allows us to quickly study the performance of different systems. An important empirical finding from extensive studies is that the optimal degree distribution seems to be of the single-degree type. This means generating the repair packets with a fixed degree instead of a true degree distribution for any given system parameters. Currently we do not have a proof for this result. However, the conjecture is further supported by a gradient-type analysis showing that the single-degree distribution is at least a local optimum.

For instance, when $w = 15$, $p = 0.1$ and $P_r = 0.2$ the optimal degree distribution seems to be $\Omega_{10} = 1.0$ which gives $Q_r = 0.0372$. When we move a small amount of probability mass to other degrees in turn, i.e. $\Omega_i = 0.1$ and $\Omega_{10} = 0.9$, the resulting performance is worse for all $i \neq 10$ demonstrating the local optimality.

All further simulation results are obtained using the single degrees giving the best analytical erasure correction performance for each set of parameters.

C. Simulation Results

We proceed by comparing the erasure correction performance using different decoding variants introduced in Sect. II-D with independent packet loss model. In Fig. 3 we

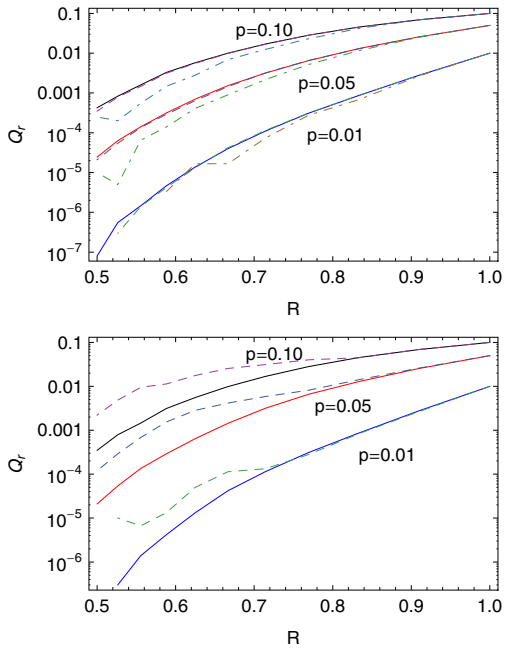

Figure 3. (Top) Comparison of simulation results with variant I (solid line) II (dashed) and III (dot-dashed). Variant III gives clearly better performance compared to I and II. (Below) Comparison of previous results [6] (dashed) to variant II (solid). Window size $w = 15$ is assumed throughout.

depict the results using decoding variants I (solid line), II (dashed line) and III (dot-dashed line) for $p = 0.01, 0.05$ and 0.10, from bottom to top, respectively, with $w = 15$. The degree distributions used are of single-degree type. The degrees were optimized using the analytical model and are presented in Table II. In addition, the table shows the optimal degrees found by an exhaustive search with simulations. As is seen from the table, the degrees differ at most by one (two for $p = 0.15$). The performances are almost identical as illustrated in Fig. 2.

The difference between the residual error rates with variants I and II is insignificant but the difference between variants II and III is perceivable. The more complete buffer handling gives extra robustness especially for lower code rates and higher channel erasure probabilities. Nevertheless, the difference is not orders of magnitude, and it is rather an implementation decision to make a trade-off between simplicity of decoding versus slightly better performance.

The results clearly indicate that an order-of-magnitude improvement in the residual loss probability can be achieved by lowering the code rate. For $w = 15$ and $p < 0.05$ this happens with code rates between 0.75 and 0.85, and for $0.05 < p < 0.10$ with code rates between 0.68 and 0.75. Further, a comparison to pure repeat code, where a systematic packet is always resent, $P_r = 1.0$, and $p = 0.01$, gives

Table II
OPTIMAL DEGREES GIVEN BY THE ANALYSIS (DEGREES OBTAINED BY EXHAUSTIVE SEARCH FROM SIMULATIONS WITH VARIANT I ARE IN PARENTHESIS).

P	$p = 0.01$	$p = 0.05$	$p = 0.10$	$p = 0.15$
0.1	15 (15)	13 (14)	10 (10)	8 (7)
0.2	15 (14)	12 (12)	10 (9)	8 (7)
0.3	13 (13)	11 (11)	9 (9)	8 (6)
0.4	12 (12)	10 (10)	9 (9)	8 (6)
0.5	11 (12)	10 (10)	9 (8)	8 (7)
0.6	11 (12)	9 (9)	9 (8)	8 (7)
0.7	10 (11)	9 (9)	9 (8)	8 (7)
0.8	10 (9)	9 (9)	8 (7)	8 (7)
0.9	10 (8 – 10)	9 (8)	8 (7)	8 (7)
1.0	9 (7 – 13)	8 (9)	8 (7)	8 (6)

$Q_r = 1.0 \cdot 10^{-4}$ for the repeat code, whereas the presented method results in $Q_r = 1.0 \cdot 10^{-7}$.

The erasure correction performance depends also on the window size. The larger the window is, the higher the probability that a specific block is repaired at some point. Figure 5 depicts the analytical performance results for window sizes $w = 5, 10, 15$ and 20 when $p = 0.07$. However, larger window sizes also result in a larger average degree (3), thus costing some computational effort.

The exact analysis used in this work is further compared to our previous method in [6], where an independence assumption in the analysis was used to calculate optimal probabilities for each window position to be included in a repair packet. The resulting strategy was always to include d most recent blocks in a repair packet. In the case $w = 15$ we found the optimal d to be 15, except for $p = 0.1$ and $P_r = 0.1, 0.2$, when it was $d = 9$ and 10, respectively. The decoding method used in [6] is the same as variant II in this work. Figure 3 depicts the performance differences for $p = 0.01, 0.05$ and 0.10. The approximations made in the previous method result in substantially higher degrees and a different repair packet strategy with a clearly inferior performance compared to the present method.

D. Simulations with Burst Erasures

For correlated loss model the simulator is modified to check possible state transition between a good and bad state, as described in Section II-E after every sent packet. Simulations were performed for situations where the duration of a bad channel state on average is either short, less than half of the window length, or long, roughly two times of the window length. We use $r = 0.15$ and 0.03 for the probability of returning to good state, thus the average length of bad channel state is either $1/r \approx 6.7$ or 33.3 slots, respectively.

Further, we evaluate scenarios where bad state corresponds to a packet loss of $p_b = 0.25$, notably larger than in the good state, and with $p_b = 1.0$ corresponding to burst-only errors in the bad state. These scenarios simulate environments where losses occur either in clusters near to each other or only in bursts of some length. Table III summarizes the used parameter

Table III
USED PARAMETER SETS IN GE MODEL SIMULATIONS

set	p_a	p_b	q	r
1	0.05	0.25	0.05	0.15
2	0.01	1.0	0.015	0.15
3	0.01	0.25	0.018	0.03
4	0.00	1.0	0.033	0.03

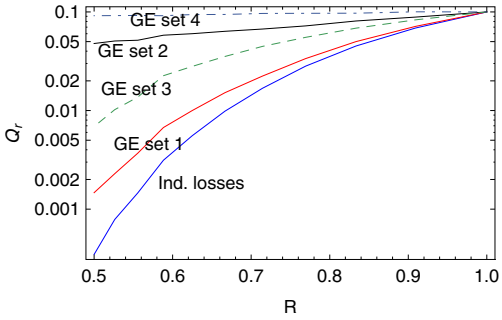

Figure 4. Simulations with GE model using sets listed in Table III and independent losses with $p = 0.1$. Window size $w = 15$.

sets for simulation, the average packet loss (7) with all sets is 0.1.

In Fig. 4 we have plotted simulations with decoding variant II using the GE model and compared them with simulations with independent loss model where $p = 0.1$. Again we have used single degree distributions optimized for independent packet losses. We see that the performance deteriorates compared to simulations with independent losses. For burst only-type loss patterns the repair strategy does not work very well. When the average length of a burst is around half-window, some repair capability is conserved. For long consecutive packet drops only minor portion of the lost packets can be recovered.

When losses are clustered but not necessarily consecutive (sets 1 and 3), the code maintains more of its repair capability and remains a viable method for erasure correction. However, we should note that the degree distributions were not optimized for bursty losses and arguably better results could be obtained if the method were extended to cope with such loss patterns. For example, when $w = 15, P = 0.3$ and $p = 0.1$ the analysis tells that the best performance is achieved by using degree-9 repair packets, with simulated $Q_r = 0.028$ (independent losses). However, if single-degree distribution is used with GE model and set 2 in Table III, best performance is achieved with degree 6 with $Q_r = 0.067$, compared to $Q_r = 0.072$ with degree 9.

E. Robustness

In order to study the robustness of the method against errors in the estimated erasure probability, we ran simulations using the degrees optimized for channel loss estimations

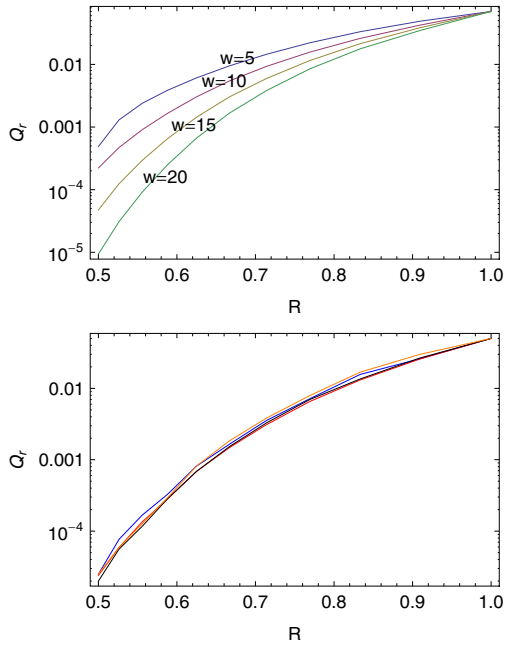

Figure 5. (Top) Comparison of the effect of different window sizes w. By increasing the window size, the correction performance can be increased. (Below) Comparison of simulations using degrees optimized for wrongly estimated channel loss probabilities (estimations $0.01, 0.05, 0.10$ and 0.20) when true $p = 0.05$.

$p = 0.01, 0.05, 0.10$ and 0.20. The resulting code rate vs. residual error probabilities are plotted in Fig. 5 for a channel with the true channel erasure probability $p = 0.05$. The results indicate that an estimation error has a negligible effect on the performance.

V. CONCLUSIONS

We have presented a fountain-coding-like erasure correction method for traffic with real-time requirements. Our concept uses a sliding window which specifies the range of current, non-expired streaming data. The code is systematic and repair packets are sent probabilistically after each systematic packet. The code is not rateless as true fountain codes are. Nevertheless, if channel conditions change the code rate can be modified on the fly via the repair packet sending probability P_r, and the optimal sending strategies can be pre-computed.

We made an exact analysis of the method using a Markov chain model for the pattern of the missing blocks in the window. The number of allowable losses was restricted to three for computational tractability. A comparison with simulation results proved the analysis accurate enough for small window sizes and realistic channel loss probabilities. An important finding is that the best performance for the proposed erasure

correction method is achieved using single-degree distributions.

The results further indicate that by lowering the code rate an order-of-magnitude improvement for channel loss probability is possible. The required code rate reduction depends on the channel loss probability p. Although an estimate for the channel loss probability p is needed in order to derive the optimal degree distribution, the proposed method was found to be quite robust against estimation errors.

Simulations using the Gilbert-Elliott burst loss model suggest that the correction capability depends on the exact characteristic of the loss parameters. For multiple consecutive packet drops the method turns out to be inefficient. However, the erasure correction performance is retained better with more conservative burst loss statistics. An extension to the presented model and optimization taking into account the possibility of correlated losses would result in better degree distributions and erasure correction performance.

A drawback of the proposed method is that it only works with fixed length packets, thus many popular audio and video compression schemes are ruled out as applications. An extension allowing variable sized packets and blocks is not trivial as the repair packet generation relies on summing up blocks of equal size bitwise together. Making all of the blocks the same size artificially, for example by padding, would result in extra overhead and inefficiency.

Future work could address the exact analysis of the more complete decoders. Also the applicability of the presented method to different applications and their requirements is an interesting research question.

REFERENCES

[1] Mattia C.O. Bogino, Pasquale Cataldi, Marco Grangetto, Enrico Magli, and Gabriella Olmo. Sliding-window digital fountain codes for streaming of multimedia contents. In *IEEE International Symposium on Circuits and Systems*, May 2007.

[2] J. Byers, M. Luby, and M. Mitzenmacher. A digital fountain approach to asynchronous reliable multicast. *IEEE Journal on Selected Areas in Communications, 20(8)*, October 2002.

[3] E.O. Elliott. Estimates of error rates for codes on burst-error channels. *Bell Systems Tech. Jouranl*, 42:1977–1997, September 1963.

[4] E.N. Gilbert. Capacity of a burst-error channel. *Bell Systems Tech. Jouranl*, 39:1253–1266, September 1960.

[5] Michael Luby. LT codes. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 271–282, 2002.

[6] Tuomas Tirronen and Jorma Virtamo. Finding fountain codes for real-time data by fixed point method. In *Proceedings of International Symposium on Information Theory and Its Applications, ISITA2008*, pages 1244–1249, Auckland, New Zealand, December 2008.

[7] Dapeng Wu, Y.T. Hoy, Wenwu Zhu, Ya-Qin Zhang, and Jon M. Peha. Streaming video over the internet: Approaches and directions. *IEEE Transactions on Circuits and Systems for Video Technology*, 11:282–300, March 2001.