Publication 4

Tuomas Tirronen and Jorma Virtamo. 2008. Greedy approach for efficient packet erasure coding. In: Proceedings of the 2008 5th International Symposium on Turbo Codes and Related Topics. Lausanne, Switzerland. 1-5 September 2008, pages 344-349.

Greedy Approach for Efficient Packet Erasure Coding

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 packet level erasure coding scheme where packets are generated using a greedy strategy. The coding is started with a systematic round of the original file blocks. This gives footing for greedy packet generation, where the degree of each randomly generated packet is determined so as to maximize the probability for the packet to release a new decoded block immediately upon arrival at the receiving end. The coding scheme relies on the estimate of erasure probability of the channel, which is used by the sender to update the belief on the number of blocks the recipient has thus far decoded. The resulting greedy sequence of packet degrees is investigated by simulations to estimate the expected number of packets needed for decoding. The simulations exhibit low overheads and further indicate that the use of greedy sequence of packet degrees provides a way to perform efficient and practical packet erasure coding. We also discuss the robustness of the scheme along with some other practical considerations.**

I. Introduction

Recently there has been increasing interest in packet based erasure correction methods which do not rely on resending missing packets but instead offer methods for correcting packet drops via coding methods. This kind of codes belong to FEC (Forward Error Correction) codes [1], which are traditionally employed on the link layer. However, modern computing systems are capable enough to perform packet based coding on the transport layer and above, and this makes it viable to consider packet based FEC coding for data transmission.

One category of this kind of codes are the so-called fountain codes [2], pioneered by M. Luby et al. [3]. Examples of these include LT codes [4] and Raptor codes [5]. Fountain codes are state-of-the-art, asymptotically optimal codes making highly scalable data transfer possible with minimal or no utilization of backwards channel. The symbols are generated stochastically in identical way, and the codes are rateless providing virtually infinite number of possible encoding symbols. Further, the performance is independent of the channel properties. However, some extra symbols (overhead) are needed for successful decoding. A design strategy for fountain codes is thus to achieve high probability for decoding with low overhead.

Our focus on this paper is to motivate and design a fountain coding inspired systematic coding scheme which meets the design goals of low overhead and reasonably simple and fast operation. A starting point in our study is the assumption that some properties of the channel are known, and we try to utilize this knowledge as much as possible. We propose a systematic method, where the data are first sent intact in consecutive packets and after that FEC symbols are generated probabilistically trying to optimally fill the gaps left due to packet erasures. The motivation for the systematic round lies in the fact that given a reasonably low channel loss probability many of the packets go through in the first round, and sending all the blocks in a deterministic way avoids wasting capacity by sending redundant information. With a carefully designed scheme a relatively low number of extra packets will be needed to fill all the gaps and to enable decoding of the original data. The received packets are used in an iterative decoding process, similar to that in the LT decoding process [4].

The packets used as encoding symbols are linear combinations of original file blocks. An exhaustive search of all possible linear combinations and optimization of the packet composition is computationally unfeasible, as the number of linear combinations for a problem of size k is 2^k. Thus, we propose a method for greedy optimization of the packet composition, which itself constitutes a sequential decision problem. The encoding process evolves by the sender updating his belief of the state of the receiver and the greedy strategy takes advantage of this.

Similar strategies have been considered for example in [6] where the authors present RT-Oblivious erasure correction using similar kind of design strategy. The so-called Growth Codes [7] for sensor networks also optimize packet degrees using similar usefulness criterion as in the proposed method.

Our method differs from many of the previously presented methods in that we design a systematic code and assume knowledge of the channel erasure probability and directly optimize the code for the given channel leading to improvement in performance. However, as the method depends on the loss statistics, it is not a true fountain code in the strict sense. As the code is optimized for a specific loss rate, we are also interested in the performance of the code in different conditions. We will address robustness by presenting some simulation results for codes optimized for different channel erasure probability than the prevailing one.

The rest of the paper is organized as follows. In section II

*The work of Tuomas Tirronen was financed by Graduate School in Electronics, Telecommunications and Automation, GETA and partly by ABI project funded by TEKES.

978-1-4244-2863-2/08/$25.00 © 2008 IEEE

we present the basic assumptions and the general structure of the encoding and decoding procedures. Then we move to motivating our optimization goals and the combinatorial calculations in section III. The performance of the proposed method is studied by simulations in section IV and the findings are discussed in section V. Finally we conclude in section VI.

II. Basic Assumptions

The scenario we consider is a point-to-point file transfer over a network using packet based FEC coding. The network is modeled as an packet erasure channel, with independent packet drops, due to congestion or bad links, with probability p.

Without loss of generality, we assume that the original file is divided into k equisized parts called blocks. First the blocks are sent in a consecutive stream as if the file was to be sent using UDP protocol [8]. One systematic round is made sending the packets intact, after which the encoding symbols are linear combinations in GF(2) of the original file blocks, trying to fill the gaps left by the initial round. The definition of a packet degree is the number of blocks that are included, or combined, in one packet. We work with bits, thus the field of symbols is $\{0, 1\}$ and the addition operation is simply XOR used bitwise for the blocks to be combined.

The steps to form a packet are then the following:

1) Determine a packet degree d.
2) Pick uniformly at random d blocks from the original file.
3) Combine the selected blocks bitwise using XOR into a packet.
4) Add the possible extra packet headers and send the packet.

Note that in step 4 above we do not explicitly make reference to any specific transportation protocol.

The decoding process is similar to LT decoding [4], where the degree-1 packets play a key role. In LT decoding, every received degree-1 packet is iteratively subtracted from the set of the already received packets and possible further decoded blocks are consequently used as degree-1 packets to keep the decoding process going on.

In our scheme, the initial round gives a high number of original file blocks unless the channel loss probability is extremely high. Note that these sent file blocks are degree-1 packets. After the initial round, if the received packet is a combination of some blocks yet unknown to the receiver, the receiver first subtracts all known blocks from the packet. The subtraction operation is again bitwise XOR. This possibly yields a new, yet unknown block. If not, then the receiver has a packet which contains only unknown blocks and the packet is saved for possible further processing, which can start if any of the included blocks in this packet becomes available later in the decoding process. To sum up the decoding procedure:

1) The receiver keeps the set of already decoded blocks and packets which have not yet been fully decoded.
2) Upon receiving a packet, subtract every already known block using bitwise XOR.

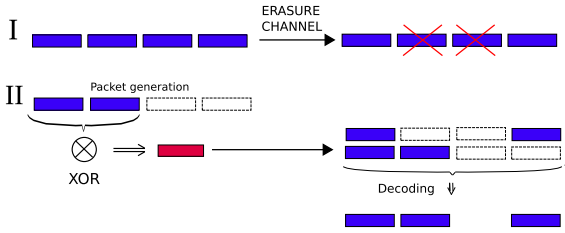

Figure 1. Overview of the coding process.

3) If the subtraction yields a new block, add it to the list of decoded blocks and check if any of the saved packets contain this block.
   a) Process all of the saved packets containing the new block, possibly revealing even more blocks.
4) If the number of decoded blocks is k, stop, otherwise go to 1.

An overview sketch of the operation is presented in Fig. 1.

The research question we address in this paper is how to determine a sequence of packet degrees d for packet generation. We need an optimization goal and method for determining the degree sequence. These are discussed in the next section, where we define our greedy optimization strategy.

III. Greedy Algorithm for Degree Sequences

A greedy algorithm can be defined to be an algorithm which "always makes the choice that look best at the moment" [9, ch. 16]. The basic idea in greedy algorithms is to make a locally efficient choice among some feasible set and ultimately hope for reaching a globally optimal solution.

The importance and characteristics of degree-1 packets leads us to define the following goal for the greedy algorithm we are constructing: For every generated packet we maximize the probability that it will contain exactly one block from the yet undecoded ones, the rest of the blocks being from the set of those already decoded. A degree optimized using this goal will give the highest probability for the receiver to immediately decode a new block.

We make the decision and optimization based on the probabilities of the composition of the receiver's current set of decoded blocks. Thus, at each step of our algorithm, we also need to update the probability distribution (the sender's belief) of the number of decoded blocks at the receiver's end.

For simplicity, in the updating algorithm we ignore the blocks discovered from the not-instantly-decodable packets that are in reality saved and may later release decoded blocks in the course of the process. This makes the algorithm slightly less optimal but it can be argued that the degradation is not severe (ultimately verified by simulation experiments), while the gain in simplicity is big. We emphasize that the simplification only pertains to the decision algorithm; in the simulations by which the performance of the designed codes are evaluated the decoding is realized in full detail.

We assume the channel erasure probability p is known by the sender. The sender can update his belief on the state of the receiver using this information. More precisely, let $f_t(n)$ be the probability that at time step t the receiver is missing n out of the total k blocks of the original message. For convenience, we count time from the end of the initial round, i.e., time $t = 0$ represents the time after the round has been completed; thereafter t actually counts the number of sent extra packets. Thus, at $t = 0$, with independent packet drops, the number of lost packets is binomially distributed,

$$f_0(n) = \binom{k}{n} p^n (1-p)^{k-n}. \quad (1)$$

Proposition 1. *Probability for decoder receiving a randomly generated degree-i packet to successfully decode one new block given n missing blocks is*

$$P(i,n) = \frac{\binom{n}{1}\binom{k-n}{i-1}}{\binom{k}{i}} = \frac{n\binom{k-n}{i-1}}{\binom{k}{i}}.$$

Proof: Exactly one of the combined blocks has to be from the n-set of yet unknown ones. The rest $i-1$ blocks have to be picked from the $k-n$ already decoded blocks. Total number of possible degree-i packets is $\binom{k}{i}$ and the result follows using basic combinatorics. ∎

Using Proposition 1 we get the unconditional probability for degree-i packet to give the decoder a possibility to decode at least one new block at time t as

$$\mathcal{P}_t(i) = \sum_{n=0}^{k} P(i,n) f_{t-1}(n). \quad (2)$$

In the greedy approach, the decision on the degree i is obtained by maximizing (2) with respect to i,

$$i^* = \arg\max_i \mathcal{P}_t(i). \quad (3)$$

The sender needs to update his belief about the number of decoded blocks the receiver is missing, i.e., the distribution $f_t(n)$ has to be updated. Now $P(i^*, n)$ gives the probability that a packet can be used to decode at least one block given n missing blocks provided that the packet was not lost in the channel. Consequently, the probability that the tth extra packet after the initial round, given n missing blocks, gets through and is useful, decreasing the number of missing blocks by one, is $(1-p)P(i^*, n)$. The complement of this expression, $1-(1-p)P(i^*, n)$, gives the probability for the event that the packet is not of the desired form or the channel drops it. If either of these events happens the number of missing packets remains unchanged.

This leads to the following updating rule

$$f_t(n) = \big(1-(1-p)P(i^*,n)\big) f_{t-1}(n) + (1-p)P(i^*,n) f_{t-1}(n+1) \quad \forall n = 0,\ldots,k \quad (4)$$

Now, with given k and channel loss probability estimate p we can run through equations (1)-(4) and calculate as long a degree sequence as needed for the packets to be sent after the initial round.

The derivation of the degree sequence calculation here does not take into account the packets which are not immediately useful but are stored for possible later use. These kind of packets would include two or more yet undecoded blocks. At this point we have kept the derivation and analysis simple and leave possible further improvements for future research.

Note that the sequence can be calculated in advance given the file size k and the channel erasure probability p, provided that the channel conditions are stable. If the erasure probability fluctuates heavily, then the sequence computation should be done dynamically during the transfer process, also updating p by an appropriate estimation algorithm.

A. Discussion

In Fig. 2 we have plotted the probabilities $\mathcal{P}_t(i^*)$ for packets sent after the initial round for $k = 100$ and the channel loss probability ranging from $p = 0.01, 0.02, \ldots, 0.10$. We see that the probability for the initial packet to be helpful is the approximately the same for all different p and is near to $1/e \approx 0.368$. This is even more evident when k is higher. The reason for this is the fact that when k is high and p low enough it is known that a Poisson distribution can be used to approximate binomial when the expectation of both of these distributions is kp.

When we generate a packet of degree i we want to make sure that the packet covers only one missing block. The number of missing blocks which fall within a randomly generated packet of degree i is binomially distributed with expectation $i \cdot p$. Now we want to make sure that $i \cdot p = 1$ and by approximating with Poisson distribution with parameter 1 it is clear that

$$\mathcal{P}(\text{a packet contains one unknown block}) = \frac{1^1}{1!} e^{-1} = 1/e.$$

Starting from $1/e$, the probabilities decrease depending on the channel loss probability. The decline results from the fact that the decoding process gets finished at some point, whereafter no packet is useful anymore. Thus the weight of the term with $n = 0$ in (2) increases resulting in declining probabilities as the process evolves. Intuitively, the higher the loss probability is, the longer the probability for a good packet stays near to $1/e$ during the process.

A related Fig. 3 shows the expected number of decoded blocks during the process with optimal degree sequences. Again, our model only considers decoding using immediately useful packets and possibly buffered packets are not taken into account here. The relationship between Figs. 2 and 3 is clear: as the expected number of decoded blocks increases, also as the probability for completed decoding increases and therefore the probability for generating a new good packet decreases.

B. Computational Complexity

The computational complexity of the described algorithm is $O(k^3)$ in the worst case. First we sum up k products in (2), thus for each degree i the complexity is $O(k^2)$ assuming that (1) has linear complexity, finally the maximum can be found by computing all k probabilities. However, the

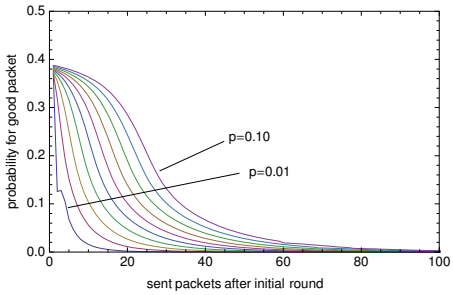

Figure 2. Evolution of probability for a usable packet during sending. Only immediately useful packets are considered.

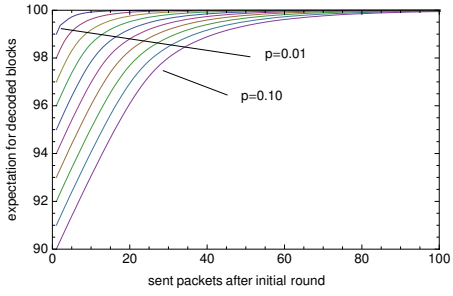

Figure 3. Expected number of decoded blocks during sending. The processing of buffered packets is not taken into account.

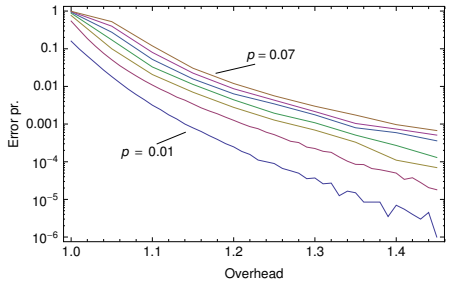

Figure 4. Decoder error probabilities for $k = 100$ and $p = 0.01, \ldots, 0.07$ as the function of received overhead factor.

Table I
SIMULATIONS FOR $k = 100$ WITH GREEDY SEQUENCES

	Received		Sent	
p	Mean	Std	Mean	Std
0.01	100.4	1.36	101.5	2.12
0.02	101.5	2.40	103.5	3.43
0.03	102.4	2.84	105.6	4.03
0.04	103.4	3.35	107.8	4.62
0.05	104.5	3.83	110.0	5.23
0.06	105.4	3.99	112.2	5.43
0.07	106.4	4.23	114.4	5.74
0.08	107.5	4.49	116.8	6.01
0.09	108.4	4.67	119.1	6.19
0.10	109.4	4.85	121.5	6.33

maximization problem has only one local optimum and the objective function is unimodal, thus it is also possible to use a dichotomic search [10] over integer domain to search for the optimum. Then the total complexity of the algorithm would be $O(k^2 \log k)$.

Updating f takes only linear time. Further, we can modify the initial loss distribution by setting the low probabilities equal to zero and thus considering a truncated distribution, requiring much less than k operations in (2). In the extreme case we can optimize for the maximum likelihood estimation of dropped packets $n = pk$ and setting $f_0(n) = \delta(n)$, where δ is Dirac's delta function. Given degree i, Proposition 1 can be used to compute $P(i, n)$ for all n in linear time, thus also the calculation of (2) is linear. Using the maximum likelihood estimators we lose some accuracy on our sequence but have an algorithm with the worst case time complexity $O(k \log k)$.

IV. NUMERICAL RESULTS

Now we study the performance of the algorithm with regard to the number of overhead packets we need in addition to k, which naturally is the absolute minimum number of packets the recipient needs to collect. We have implemented a simulator which takes a degree sequence as a parameter and goes trough the actual coding and decoding processes and drops packets independently with a given channel loss parameter p. During the simulations, where the degree sequence given by our greedy algorithm is used as input, we collect statistics of the mean value and standard deviations of the number of the packets needed for decoding. Observations are discussed in section V.

A. Simulations With Greedy Sequences

In Table I we have the mean overheads from 10000 simulations with optimized degree sequences for $k = 100$ with loss probabilities $p = 0.01, 0.02, \ldots, 0.10$. We use precomputed sequences with length 100, long enough to guarantee decoding in the simulated scenarios.

The performance is further depicted in Fig. 4 where the decoder error probability is plotted for $k = 100$ as a function of the overhead. Decoder error is declared if the decoding is not successful with specific amount of received overhead. The results were obtained by simulations stopped at predetermined number of received overhead packets and calculating the number of cases where decoding was successful at the point. For comparison, with 5% overhead the probability for decoding is approximately $7 \cdot 10^{-4}$ for $k = 1000$ and little less than 0.02 for $k = 100$.

In Fig 5 we have plotted the calculated greedy degree sequences for $p = 0.01, 0.05, 0.09$ and 0.12, with $k = 100$.

In real life, it is seldom the case that the channel loss probability is exactly known or stays the same during a data

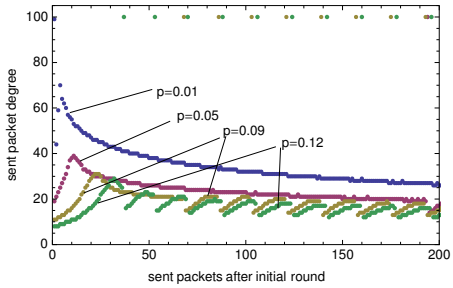

Figure 5. Greedy degree sequences for $k = 100$ and different values for p.

Table II
COMPARISON OF STATISTICS WHEN THE SEQUENCES HAVE BEEN OPTIMIZED FOR p DIFFERING FROM TRUE CONDITIONS

	$p = 0.01$		$p = 0.05$		$p = 0.09$	
True p	Mean	Std	Mean	Std	Mean	Std
0.01	1010	5.2	1077	20	1149	38
0.02	1035	36	1079	16	1160	30
0.03	1142	97	1075	14	1160	26
0.04	1343	171	1068	13	1156	23
0.05	1636	260	1059	13	1149	21
0.06	1996	285	1051	13	1141	20
0.07	2252	189	1048	31	1133	19
0.08	2354	83	1080	92	1124	18
0.09	2383	41	1194	185	1114	18
0.10	2396	38	1404	280	1105	17

transfer. Thus we are also interested in how a particular degree sequence performs if the true channel loss probability differs from the value of p the degree sequence was optimized for. As an example case we take the degree sequences optimized for $p = 0.01, 0.05$ and 0.09 and run simulations for channel loss probabilities $p = 0.01, 0.02, \ldots, 0.10$. The file size was $k = 1000$ blocks. The results are given in Table II.

Instead of using (1) for the initial distribution for the number of missing blocks after the initial round we can adopt a simplified approach of using a maximum likelihood estimation for the number of missing blocks and use that as a starting point for the senders belief about the receivers state. This way we do not update a distribution of possible states of the receiver but use a simple point estimation at every step for the number of blocks the receiver still needs for decoding. The impact of this change is that the resulting algorithm case can be implemented in a way which yields worst case time complexity of $O(k \log k)$.

V. DISCUSSION

The results presented in previous section indicate that the presented method yields reasonably low mean packet overheads in the coding process. The complexity of calculating the degree sequences is low, and adjustable to prevailing circumstances providing a trade-off between overhead and computational complexity.

Table III
MEANS AND STANDARD DEVIATIONS FOR NUMBER OF PACKETS NEEDED FOR DECODING FROM 10000 SIMULATIONS FOR $k = 100$ WITH GREEDY SEQUENCES USING SIMPLE BELIEF

	Received		Sent	
p	Mean	Std	Mean	Std
0.01	101.2	5.48	102.3	6.05
0.02	102.4	6.69	104.5	7.57
0.03	103.7	8.19	107.0	9.31
0.04	104.8	8.14	109.2	9.51
0.05	105.9	8.37	111.4	9.90
0.06	106.7	7.89	113.5	9.54
0.07	107.7	7.61	115.8	9.40
0.08	108.7	7.75	118.1	9.63
0.09	109.8	7.89	120.7	9.86
0.10	110.6	7.63	122.8	9.63

A. Numerical Results

Table I and Fig. 4 asserts that the overheads are quite low and vary with the channel loss probability. In particular, the overhead is clearly smaller than with the LT coding for $k = 100$ [11]. The number of received packets is roughly $1 - p$ times sent packets as it should be. The correlation to the channel erasure probability seems to be linear, which is natural as the expected number of lost packets during the initial round is kp. The standard deviations are a few percentages of the mean, indicating reasonably robust operation and thus tolerable worst case results. The used application naturally defines what the true tolerance limits are. The performance improves with growing k.

1) Qualitative Nature of the Sequences: Some qualitative properties of the obtained degree sequences can be pointed out in Fig. 5. First of all, the average degree of the whole sequence seems to be near to $k/5$ for the tested parameters. The sequences start from a degree lower than $k/5$, roughly from the value $1/p$, then the degrees grow a little bit and form a peak, after which the degrees decrease and ultimately starts to do a sawtooth like pattern. The location of the peak depends on the channel loss probability p; the higher p the later in the sequence the peak occurs.

The sawtooth phase consists of teeth, each comprising approximately ten packets with almost linearly increasing degrees, followed by a single degree-k packet. This means that every once in a while a packet which combines all of the source blocks is sent. This kind of packets guarantee decoding when the decoder has all but one blocks. With growing p the sawtooth phase begins earlier in the sequence.

2) Robustness of Greedy Degree Sequence: When the channel erasure probability is different from p used in the algorithm, the results are suboptimal. In Table II we have figures of this kind of scenario. We see that in practice for loss probabilities lower than the one the sequence is optimized for, the results are quite acceptable. However, for higher true loss probabilities the overheads grow quickly. Thus a viable

strategy would be to be a little pessimistic about the true erasure probability and run the optimization for a slightly higher p to be on the safe side.

3) Using Maximum Likelihood Belief: In Table III we compare simulation results between the log-linear complexity scheme and the full algorithm. We note that compared to Table I the overheads are roughly one packet worse. The standard deviations, however, are considerably larger thus indicating that using the simpler calculations results in fluctuating performance. Depending on the application, user can choose between a very fast degree calculation with an inferior worst case performance or a more extensive algorithm with better controlled deviations from the mean.

4) Overheads: Overall, the results in Tables I and II suggest that the overhead for correctly optimized degree sequences are roughly $p \cdot k$. Due to lack of space we do not present more extensive results for varying k but these simulations reinforce this conclusion.

5) Burst Errors: In reality the erasures are often correlated or arrive as bursts. The presented method is derived assuming independent losses, thus the presented scheme could probably be improved for taking into account correlated loss statistics. However, if the lengths of typical bursts are shorter than the file size, we argue that our method approximates the optimal method reasonably well. If the burst happens on the initial round and the code is optimized for correct p the performance should not deteriorate at all. If there is no burst in the systematic part there is no decoding left to be done and the decoding will succeed with probability 1 with no overhead.

With multiple bursts during sending the performance generally depends on the length of the bursts. If p gives the proper average error probability, the coding should work reasonably well. This conclusion reinforced by some brief simulations not presented in this work. Further results and studies of bursty losses are left for possible future research.

B. Integration with DCCP

The presented method relies on the information about the erasure probability in order to calculate the degree sequences. Thus in order to implement a working protocol employing our strategy one needs some way of conveying this information back to the sender. We suggest using DCCP (Datagram Congestion Control Protocol) [12] for this. In brief, DCCP is a transport protocol which provides congestion control for unreliable datagram flows. DCCP can be used with different congestion control schemes, in our scenario using CCID-2 [13] would provide TCP-like congestion control with an option for ack ratio, i.e., how many packets are acknowledged by one acknowledgement packet. This could be used to improve the robustness of the method by lowering the ack ratio for channels with much variation. Naturally, with this kind of operation relying on feedback we even further sacrifice on the principle of true fountain coding.

VI. Conclusions

We proposed a systematic packet erasure coding scheme which makes use of the estimate of the channel erasure probability. The transfer is started with a full round of original file blocks, which provides a starting point for the greedy packet generation method. The greedy combination process provides low packet overheads and the degree sequence calculation is efficient. The distinction to other comparable code types, e.g., the LT or Raptor codes, is the exploitation of the estimate of the erasure probability, with a belief of the current receiver state. Consequently our scheme is not channel independent with regard to the overhead as the other state-of-the-art coding schemes are. However, our scheme demonstrates low average overheads for any file sizes.

The results also suggest that the estimation of the loss probability should be a pessimistic one, on the high side, in order to provide reasonable robustness of the scheme on varying channel conditions. The updating of the estimate itself was not discussed in the paper. However, by shortening the interval between the estimate updates the robustness could be arguably improved.

Future research possibilities include the extension of the current model to take into account also the packets that are not immediately useful. Also designing the code to handle directly possible bursty losses should be addressed. Further practical considerations on implementing the presented method and possible network implications could also be made.

VII. Acknowledgements

The authors would like to thank Dr. Esa Hyytiä for useful discussions.

References

[1] S. B. Wicker, *Error Control Systems fo Digital Communication and Storage*. Prentice-Hall, Inc., 1995.

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

[3] 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.

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

[5] A. Shokrollahi, "Raptor codes," preprint at http://www.inference.phy.cam.ac.uk/mackay/dfountain/RaptorPaper.pdf, cited 1.3.2005.

[6] A. Beimel, S. Dolev, and N. Singer, "RT Oblivious Erasure Coding," *IEEE/ACM Transactions on Networking*, vol. 15, no. 6, pp. 1321–1332, Nov. 2007.

[7] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein, "Growth Codes: Maximizing Sensor Network Data Persistence," in *Proceedings of SIGCOMM*, 2006.

[8] J. Postel, "User Datagram Protocol," Aug. 1980, http://www.ietf.org/rfc/rfc768.txt.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition*. The MIT Press, 2001.

[10] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 2nd ed. John Wiley and Sons, Inc., 1993.

[11] E. Hyytiä, T. Tirronen, and J. Virtamo, "Optimizing the degree distribution of LT codes with an importance sampling approach," in *Proc. The 6th International Workshop on Rare Event Simulation*, Oct. 2006, pp. 64–73.

[12] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," Internet Engineering Task Force, RFC 3452, Mar. 2006, http://www.ietf.org/rfc/rfc4340.txt.

[13] S. Floyd and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control," Internet Engineering Task Force, RFC 3452, Mar. 2006, http://www.ietf.org/rfc/rfc4340.txt.