

## MIT Open Access Articles

*On code parameters and coding vector  
representation for practical RLNC*

The MIT Faculty has made this article openly available. **Please share**  
how this access benefits you. Your story matters.

**Citation:** Heide, Janus et al. "On Code Parameters and Coding Vector Representation for Practical RLNC." IEEE International Conference on Communications (ICC), 2011. 1–5.

**As Published:** <http://dx.doi.org/10.1109/icc.2011.5963013>

**Publisher:** Institute of Electrical and Electronics Engineers (IEEE)

**Persistent URL:** <http://hdl.handle.net/1721.1/73680>

**Version:** Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

**Terms of use:** Creative Commons Attribution-Noncommercial-Share Alike 3.0



# On Code Parameters and Coding Vector Representation for Practical RLNC

Janus Heide, Morten V. Pedersen and Frank H.P. Fitzek

Faculty of Engineering and Science,  
Aalborg University, Aalborg, Denmark  
Email: [jah|mvp|ff]@es.aau.dk

Muriel Médard

Massachusetts Institute of Technology  
Cambridge, Massachusetts, USA  
Email: medard@mit.edu

**Abstract**—Random Linear Network Coding (RLNC) provides a theoretically efficient method for coding. The drawbacks associated with it are the complexity of the decoding and the overhead resulting from the encoding vector. Increasing the field size and generation size presents a fundamental trade-off between packet-based throughput and operational overhead. On the one hand, decreasing the probability of transmitting redundant packets is beneficial for throughput and, consequently, reduces transmission energy. On the other hand, the decoding complexity and amount of header overhead increase with field size and generation length, leading to higher energy consumption. Therefore, the optimal trade-off is system and topology dependent, as it depends on the cost in energy of performing coding operations versus transmitting data. We show that moderate field sizes are the correct choice when trade-offs are considered. The results show that sparse binary codes perform the best, unless the generation size is very low.

## I. INTRODUCTION

Network Coding (NC) is a promising paradigm that breaks with the existing store-and-forward paradigm in computer networks [1]. NC enables coding on the fly at the individual node in the communication network, and thus is fundamentally different from the end-to-end approach of channel and source coding. Thus packets are no longer treated as atomic entities as the number of incoming and outgoing packets per node, is not necessarily equal and data may be combined and re-combined at any point in the network. This new feature can provide advantages over traditional routing in meshed networks, and fits perfectly with the ideas of cooperative and distributed networks.

A promising popular approach, introduced in [2], is RLNC. In RLNC coding is performed at random which minimizes the need for signaling, compared to deterministic codes. Because coding is performed randomly there is a non-zero probability, that a received coded symbol is linearly dependent on already received symbols, and thus unusable. Figure 1 illustrates the benefits of coding. When no coding is used nodes can only forward symbols, and as the relays must forward two different packet for the sink to decode. If binary coding is used an additional symbol can be created,  $A \oplus B$ , and thus the probability that the relays forward two different symbols increases. If coding is performed over a higher field many more symbols can be created,  $\alpha A \oplus \beta B$ , and the probability that the relays forward two different symbols increases.

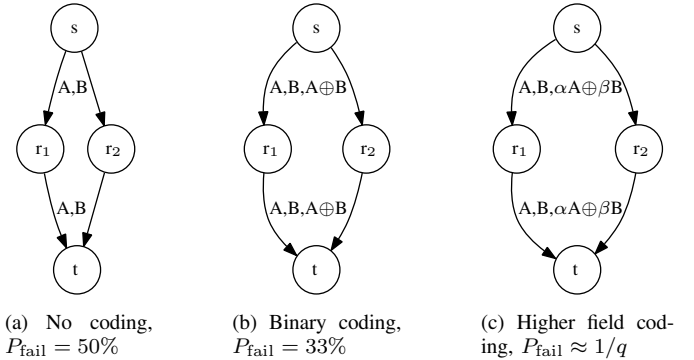


Fig. 1: Example network with and without coding,  $s$  is the source,  $t$  is the sink,  $r_1$  and  $r_2$  are relays.

This probability result in the *linear dependence* overhead. The parameters; generation size, field size and density influence this overhead, and are often assumed to be high as this decreases the probability of linear dependence. To decode a received symbol a sink needs the coding vector of the symbol, which describes the coding operation performed during encoding. This information must be included as header information when the coded symbol is transmitted, which results in an additional *header* overhead.

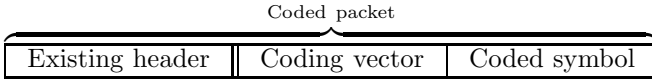
Unfortunately, when coding is performed on a computational device, the parameters also affect the coding throughput, which is the rate at which coding is performed [Mb/s]. Higher values generally result in lower coding throughput [3]–[6]. However, the coding throughput also depends on less deterministic parameters e.g. the hardware platform, programming language, and implementation optimizations. Recently [7] have shown that a systematic code with a RLNC-based redundancy can achieve low computational complexity while remaining binary, but only over a single-hop system.

The objective of this work is to increase the coding throughput without significantly increasing the linear dependence overhead. Our contribution is two fold. In Section II we analyze the impact of changing the field size, generation size, and density, and provide bounds for the resulting linear dependence overhead. In Section III we consider the representation of the coding vector, the resulting header overhead.

## II. CODING

The data, of size  $B$  that is to be transferred from a source to one or more sinks is divided into generations of size  $g \cdot m$ , a generation is sometimes also referred to as a source block or a batch. Each generation constitutes  $g$  symbols of size  $m$ , where  $g$  is called the generation size. The  $g$  original symbols of length  $m$  in one generation, are arranged in the matrix  $M = [\mathbf{m}_1; \mathbf{m}_2; \dots; \mathbf{m}_g]$ , where  $\mathbf{m}_i$  is a column vector. In an application the block of data can be a file or a part of a media stream, and is divided into  $\lceil \frac{B}{m} \rceil$  pieces, called symbols. Generation number 0 constitutes the first  $g$  symbols, or the first  $g \cdot m$  bytes of data, there are  $\lceil \frac{B}{g \cdot m} \rceil$  such generations.

To encode a new symbol  $\mathbf{x}$  from a generation at the source,  $M$  is multiplied with a randomly generated coding vector  $\mathbf{g}$  of length  $g$ ,  $\mathbf{x} = M \times \mathbf{g}$ . In this way we can construct  $g + r$  coded symbols and coding vectors, where  $r$  is any number of redundant symbols as the code is rateless. When a coded symbol is transmitted on the network it is accompanied by its coding vector, and together they form a coded packet. A practical interpretation is that each coded symbol, is a combination or mix of the original symbols from one generation. The benefit is that nearly infinite coded symbols can be created.



In order for a sink to successfully decode a generation, it must receive  $g$  linearly independent symbols and coding vectors from that generation. All received symbols are placed in the matrix  $\hat{X} = [\hat{x}_1; \hat{x}_2; \dots; \hat{x}_g]$  and all coding vectors are placed in the matrix  $\hat{G} = [\hat{g}_1; \hat{g}_2; \dots; \hat{g}_g]$ , we denote  $\hat{G}$  the decoding matrix. The original data  $M$  can then be decoded as  $\hat{M} = \hat{X} \times \hat{G}^{-1}$ . To spread the computational load this can be performed with an on-the-fly version of Gaussian elimination. In practice if approximately any  $g$  symbols from a generation are received the original data in that generation can be decoded. This is a much looser condition, compared to when no coding is used, where exactly all  $g$  unique original symbols must be collected [8].

Any node that have received  $g'$ , where  $g' = [2, g]$  is the number of received linearly independent symbols from a generation and is equal to the rank of  $\hat{G}$ , can decode. All received symbols are placed in the matrix  $\hat{X} = [\hat{x}_1; \hat{x}_2; \dots; \hat{x}_{g'}]$  and all coding vectors in the matrix  $\hat{G} = [\hat{g}_1; \hat{g}_2; \dots; \hat{g}_{g'}]$ . To decode a symbol these matrices are multiplied with a randomly generated vector  $\mathbf{h}$  of length  $g'$ ,  $\tilde{\mathbf{g}} = \hat{G} \times \mathbf{h}$ ,  $\tilde{\mathbf{x}} = \hat{X} \times \mathbf{h}$ . In this way we can construct  $r'$  randomly generated recoding vectors and  $r'$  recoded symbols.  $r' > g'$  is possible, however a node can never create more than  $g'$  independent symbols. Note that  $\mathbf{h}$  is only used locally and that there is no need to distinguish between coded and recoded symbols. In practice this means that a node that have received more than one symbol can recombine those symbols into recoded symbols, similar to the way coded symbols are constructed at the source.

### A. Generation Size

The generation size  $g$  is the number of symbols over which encoding is performed, and defines the maximal number of symbols that can be combined into a coded symbol. Data is decoded on a per generation level, thus at least  $g$  symbols must be received before decoding is possible. Hence the size of a generation  $g \cdot m$  dictates the decoding delay which is the minimum amount of data that must be received before decoding is possible.

From a *linear dependence* overhead point of view  $g$  should be high, especially in multiple-sink broadcast networks, where a low  $g$  increases the amount of expected transmissions per symbol, due to erasures [5]. From a practical point of view, decoding delay and coding throughput must also be considered. For bulk downloads the decoding delay is not important. But for streaming services and Voice over Internet Protocol (VoIP) in particular it is critical, and  $g$  must be chosen with care. Additionally a high  $g$  generally decreases the coding throughput, thus  $g$  must be chosen low enough to ensure satisfactory coding throughput on the given platform.

To achieve reliability in a practical system some signaling is necessary for each generation. A simple form could be to acknowledge when each generation is successfully decoded. Thus the benefits of NC in terms of reduced signaling diminish, when the generation size is decreased, as the number of generations necessary to represent some fixed amount of data increases. This overhead is protocol and topology dependent, and therefore outside the scope of this work.

### B. Field Size

The field size,  $q$ , defines the size of the finite field over which all coding operations are performed, and thus the number of unique field elements. A necessary but insufficient condition for decoding is that all rows have at least one non-zero scalar. This probability can be found from the probability of receiving a symbol where at least one scalar in the coding vector, that corresponds to a symbol for which the decoder has not yet identified a pivot element, is non-zero. The following bound for linear independence, when each scalar in the coding vector is drawn uniformly, is assumed in an alternative form in [9], [10] and is said to hold when  $q$  is high.

$$P_{\text{independent}} \leq 1 - \frac{1}{q^{g-g'}} \quad (1)$$

In [5] we observed the probability of generating  $g$  symbols that are not all independent, given by Equation (2), is a good approximation even at low values of  $q$ .

$$1 - \prod_{g'=0}^{g-1} \left( 1 - \frac{1}{q^{g-g'}} \right) \quad (2)$$

Thus as  $g'$  goes towards  $g$  it becomes increasingly more difficult to receive useful symbols, because the coding vector must be non-zero in at least one of the  $g - g'$  corresponding scalars. This yields the following transition probabilities.

$$P_{g' \rightarrow g'} = \frac{1}{q^{g-g'}} \quad P_{g' \rightarrow g'+1} = 1 - \frac{1}{q^{g-g'}}$$

$$\mathbf{P} = \begin{bmatrix} \frac{1}{q^g} & 0 & \cdots & 0 \\ (1 - \frac{1}{q^g}) & \frac{1}{q^{g-1}} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & (1 - \frac{1}{q^1}) & 1 \end{bmatrix}$$

Thus the expected amount of overhead for a generation can be found by evaluating the probability that the rank is not full after  $k$  transmissions,  $p(g' \neq g)$ . Initially no symbols are received and therefore the starting pmf  $\mathbf{s}$  is,  $\mathbf{s} = [1, 0, \dots]$ . When less than  $g$  symbols are received,  $p(g' = g) = 0$ , and hence the overhead can be evaluated as.

$$\alpha \geq \sum_{k=g'}^{\infty} \mathbf{p}^k(g' \neq g), \quad \mathbf{p}^k = (\mathbf{P}^k \times \mathbf{s}) \quad (3)$$

This can be rewritten to the form in Equation (4).

$$\begin{aligned} \alpha(q, g) &\geq \sum_{g'=0}^{g-1} \left( \left( 1 - \frac{1}{q^{g-g'}} \right)^{-1} - 1 \right) \\ &= \sum_{g'=0}^{g-1} \left( \frac{1}{q^{g-g'} - 1} \right) \end{aligned} \quad (4)$$

It might be expected that a decreased density would impact this directly. However as decoding progresses the not-decoded remainder of the coding vectors will go towards a uniform drawn distribution, due to the fill-in effect. Therefore a separate contribution to the overhead stems from the density.

### C. Density

The ratio of non-zero scalars in a coding vector is often referred to as the *density*. The density of a coding vector  $\mathbf{h}$  with a generation size  $g$  is defined by Equation (5).

$$d(\mathbf{h}) = \frac{\sum_{k=1}^g (h_k \neq 0)}{g} \quad (5)$$

A necessary but insufficient condition for decoding is that all columns have at least one non-zero scalar. For a generation a receiving node can have  $j = [0, g]$  non-zero columns. The probability that a scalar is non-zero in a received symbol is  $d$ . Before the transition there are  $j$  non-zero columns, after the transition there are  $j'$ . Thus the number of possible combinations for the transition is given by  $\binom{g-j}{g-j'}$ .  $j' - j$  columns becomes non-zero with probability  $d$ .  $g - j'$  columns remain all-zero with probability  $1 - d$ . Thus the probability of transition from state  $j$  to state  $j'$ , where  $j' \geq j$ , is.

$$O_{j \rightarrow j'} = d^{j' - j} \cdot (1 - d)^{g - j'} \cdot \binom{g - j}{g - j'} \quad (6)$$

$$\mathbf{O} = \begin{bmatrix} (1 - d)^g & 0 & \cdots & 0 \\ d \cdot (1 - d)^{g-1} \binom{g}{g-1} & (1 - d)^{g-1} & & \vdots \\ \vdots & & \ddots & 0 \\ d^g & \cdots & d & 1 \end{bmatrix}$$

Initially all columns in the decoding matrix consist of zero vectors. Therefore the starting pmf  $\mathbf{s}$  is,  $\mathbf{s} = [1, 0, \dots]$ . At least  $g$  symbols must be received for decoding to be possible. Hence the estimated number of symbols that must be received in addition to  $g$  before all columns contain non-zero values can be evaluated as.

$$\beta \geq \sum_{k=g}^{\infty} \mathbf{t}^k(j \neq g), \quad \mathbf{t}^k = (\mathbf{O}^k \times \mathbf{s}) \quad (7)$$

The probability that one column is the zero vector is the probability that one scalar is zero to the power of the number of received symbols. From this we can determine the probability that at least one additional packet is needed when  $k$  symbols have been received.

$$\beta(d, g) \geq \sum_{k=g}^{\infty} \left( 1 - (1 - (1 - d)^k)^g \right) \quad (8)$$

for  $0 < d \leq 1 - q^{-1}$

### D. Linear Dependence Overhead

The total overhead of a given code is given by the expected number of redundant symbols necessary.

$$\frac{\alpha + \beta}{g} \quad (9)$$

To verify Equations (4), (8), and (9) we compare with measured overhead obtained from a high number of runs of our own implementation of RLNC. The results are plotted on Figure 2, where  $g$  is on the x-axis and the resulting overhead is on the y-axis.

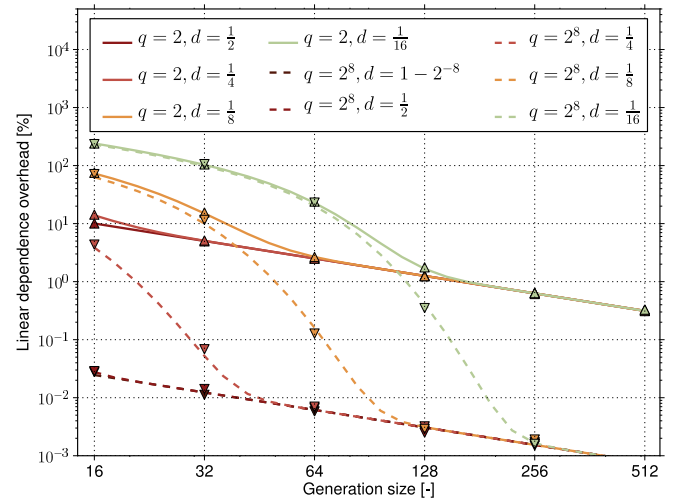


Fig. 2: Linear dependence overhead, analytical values are plotted as lines, measured values are marked with triangles.

On Figure 2, triangles denote measured overhead, which show that the analytical results are a good approximation of the measured values, the error is below 6 % for all measured settings. As  $g$  increases the overhead decreases, and when  $g$  becomes sufficiently high,  $d$  can be decreased with no penalty to the overhead.

### III. CODING VECTOR REPRESENTATION

To decode a received symbol, a node must in addition to the symbol, hold the corresponding coding vector which results in the *header* overhead. It has been suggested to use a predefined *pseudo random* function to generate coding vectors based on a seed, and then include the seed instead of the coding vector itself, e.g. in [11]. This reduces the overhead to the size of the seed, but also reduces the number of unique coding vectors to the size of the seed. This approach is not suitable for recoding [12]. The reason is that during recoding the coding vector is not drawn randomly but instead computed as  $\tilde{g} = \hat{g} \times \mathbf{h}$  where  $\mathbf{h}$  is random. As  $\tilde{g}$  can take  $q^g$  values, not all possible  $\tilde{g}$  can be constructed from the seed. Even if this was possible there is the challenge of identifying which seed produces the wanted coding vector.

We assume that recoding is a requirement, and thus the pseudo random function approach cannot be used. Instead we consider some other representations. A simple but *naive* approach is to construct the coding vector from all the scalars.

$$\begin{bmatrix} s_0 & s_1 & \dots & s_g \end{bmatrix}$$

Each scalar can be represented by  $\log_2(q)$  bits, and there are  $g$  such scalars. We denote this overhead introduced by the coding vector  $\gamma$ .

$$\gamma_1 = \log_2(q) \cdot g \quad (10)$$

If the density is low, the coding vector will be sparse, and will mostly consist of 0's. Hence the *naive* approach will be very inefficient. Instead we can represent each non-zero scalar by an index-scalar pair. It is also necessary to append the number of index-scalars pairs, as this can vary.

$$\begin{bmatrix} t & i_0 & s_0 & i_1 & s_1 & \dots & i_t & s_t \end{bmatrix}$$

The number of index-scalar pairs,  $t$ , takes up at most  $\log_2(g)$  bits, as the maximal number of non-zero scalars is  $g$ . Each index takes  $\log_2(g)$  bits and each scalar takes  $\log_2(q)$  bits, and on average there are  $g \cdot d$  such pairs. For  $q = 2$  it is only necessary to include the indices's as there is only one non-zero scalar.

$$\gamma_2 = \log_2(g) + (\log_2(g) + \log_2(q)) \cdot g \cdot d \quad (11)$$

The coding vector can also be represented by a bit array, that indicates which scalars are non-zero, and the values of these scalars.

$$\begin{bmatrix} a_0 & a_1 & \dots & a_g & s_x & s_y & \dots & s_z \end{bmatrix}$$

The bit array can be represented by  $g$  bits. Each of the scalars takes  $\log_2(q)$  bits, and on average there are  $g \cdot d$  such scalars for each encoded symbol. If the bit array is compressed with an optimal code, the amount of bits necessary to represent

it can be reduced from  $g$  to the entropy of the bit vector,  $H(a)$ , which can be calculated from  $d$  and  $g$ .

$$\gamma_3 = H(a) + \log_2(q) \cdot g \cdot d \quad (12)$$

#### A. Total Overhead

The *total* overhead constitutes the *linear dependence* and *header* overhead, divided by the size of a generation  $g \cdot m$

$$\frac{(\alpha + \beta) \cdot m + (g + \alpha + \beta) \cdot \gamma}{g \cdot m} \quad (13)$$

Three examples of the contributions to the total overhead is illustrated on Figure 3. On the x-axis in the range  $[10^{-3}, 1]$ , on the y-axis is the resulting overhead, and the minimal overhead is marked with a vertical line. On the figure, four contributions from Equation (13) are stacked,  $\frac{\alpha}{g}$  from the field size,  $\frac{\beta}{g}$  from the density,  $\frac{\gamma}{m}$  from the coding vector representation, and the remainder  $\frac{(\alpha + \beta) \cdot \gamma}{g \cdot m}$ .

On Figure 3 it can be seen that the contribution from  $\alpha$  is constant. On Figure 3a the contribution from  $\beta$  is dominating until the density reaches approximately 0.1. When  $g$  is larger, in Figure 3b, the contribution from  $\alpha$  decreases, and the contribution from  $\beta$  decreases faster. However, the contribution from  $\gamma$  becomes bigger, for high densities. For a higher  $q$ , in Figure 3c, the contribution from  $\alpha$  is significantly reduced. However, for high densities the contribution from  $\gamma$  dominates.

The interesting result is the minimal obtainable overhead for a given value of  $g$ . Therefore we have identified this for different values of  $g$ , the result of this search is plotted on Figure 4, where  $g$  is on the x-axis and lowest total obtainable overhead is on the y-axis.

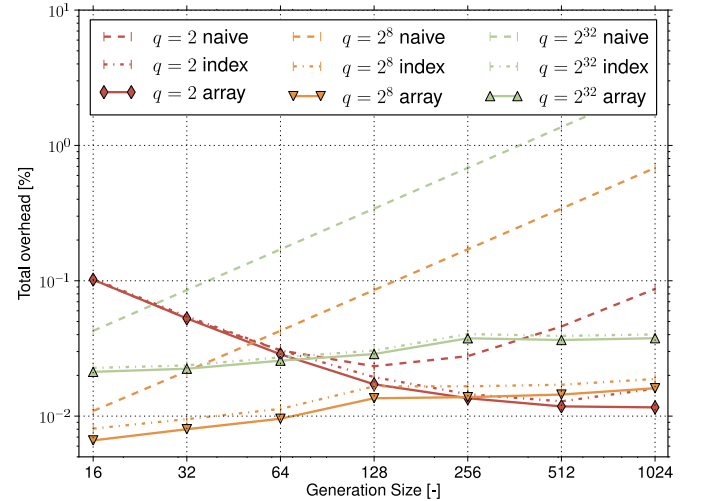


Fig. 4: Lowest total overhead obtainable for different approaches when  $g$  is varied.

Interestingly the result shows that  $q = 2^{32}$  should never be used. The reason is that the increased entropy of the coding vector is much larger compared to the benefit from the high  $q$ -value. For  $g < 256$  the lowest overhead can be obtained when  $q = 2^8$ . For  $g > 256$ ,  $q = 2$  can give the lowest overhead.

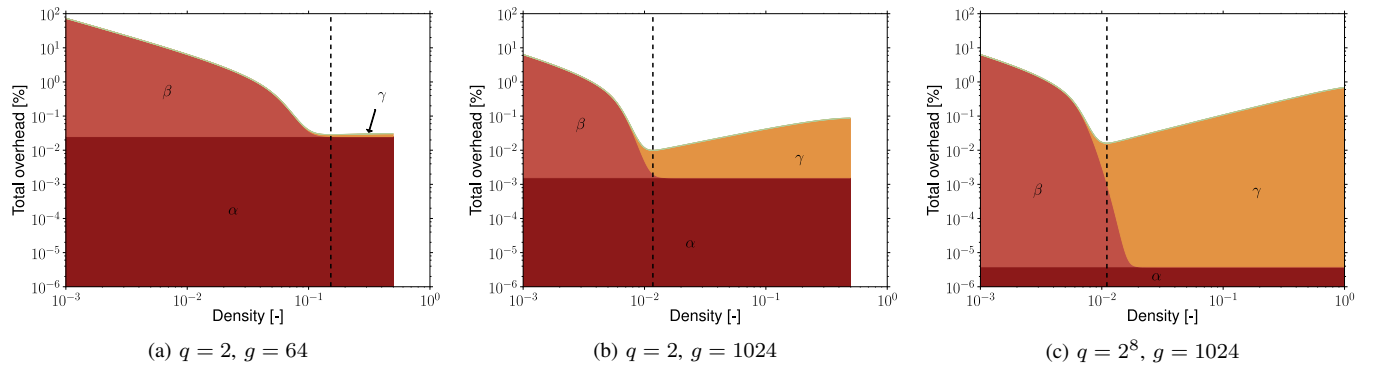


Fig. 3: Examples of the total overhead, that is a function of the field size, the density, and the coding vector representation.

One might conclude that a very low generation size would be the best choice. However, it is important to remember the consequences of a low generation size, see Section II-A. Since the *index* approach is much simpler to implement compared to the *array* approach, it may still be useful as the performance when  $q = 2$  is similar for the two approaches.

Remark that all evaluations are performed at  $m = 1500$  B. This fits well with bulk data distribution or very high rate media streaming over Wireless Local Area Network (WLAN) networks. To evaluate settings where  $m$  is significantly different, see [13] for a small script to evaluate the overhead for different setting.

#### IV. CONCLUSION

In this paper we have analyzed the transmission overhead of RLNC, as a function of the generation size, field size, density, and coding vector representation. The results have been verified with measurements from our own implementation of RLNC. The results show that generally, in the case where recoding must be supported, a field size of 2 and a low density should be used. If the field size and density is increased, the bits necessary for coding vector representation increases faster than the improvement obtained from the lower amount of linearly dependent packets. However, if the generation size is very low a larger field size than 2 provides the lowest overhead. From a transmission overhead point-of-view, if the recoding operation is not required, the generation size, field size, and density should be chosen as high as possible. However, these parameters also impact the coding throughput, therefore they must be chosen with care in practical applications. As the coding throughput is implementation and topology dependent, no single set of optimal values exists. The results in this work can be used when RLNC is deployed in a real application. The practical performance in terms of coding throughput and energy consumption of the used RLNC implementation, can be compared with the transmission overhead obtained for given parameters. Hence a good trade-off in terms of coding throughput, transmission overhead, and energy can be determined, for a given application and network topology.

#### ACKNOWLEDGMENT

This work was partially financed by the CONE project (Grant No. 09-066549/FTP) granted by Danish Ministry of Science, Technology and Innovation, and the ENOC project in collaboration with NOKIA, Oulu.

#### REFERENCES

- [1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.
- [2] T. Ho, R. Koetter, M. Médard, D. Karger, and M. Ros, "The benefits of coding over routing in a randomized setting," in *Proceedings of the IEEE International Symposium on Information Theory, ISIT '03*, June 29 - July 4 2003. [Online]. Available: [citeseer.ist.psu.edu/ho03benefits.html](http://citeseer.ist.psu.edu/ho03benefits.html)
- [3] H. Shojania and B. Li, "Parallelized progressive network coding with hardware acceleration," in *Quality of Service, 2007 Fifteenth IEEE International Workshop on*, June 2007, pp. 47–55.
- [4] X. Chu, K. Zhao, and M. Wang, "Massively parallel network coding on gpus," in *Performance, Computing and Communications Conference, 2008. IPCCC 2008. IEEE International*, December 2008, pp. 144–151.
- [5] J. Heide, M. V. Pedersen, F. H. Fitzek, and T. Larsen, "Network coding for mobile devices - systematic binary random rateless codes," in *The IEEE International Conference on Communications (ICC)*, Dresden, Germany, 14–18 June 2009.
- [6] H. Shojania, B. Li, and X. Wang, "Nuclei: Gpu-accelerated many-core network coding," in *The 28th Conference on Computer Communications (INFOCOM 2009)*, April 2009.
- [7] D. Lucani, M. Médard, and M. Stojanovic, "Systematic network coding for time-division duplexing," jun. 2010, pp. 2403–2407.
- [8] W. Feller, *An Introduction to Probability Theory and Its Applications, Volume 1*. Wiley, 1968.
- [9] A. Eryilmaz, A. Ozdaglar, and M. Médard, "On delay performance gains from network coding," *Information Sciences and Systems, 2006 40th Annual Conference on*, pp. 864–870, March 2006.
- [10] D. E. Lucani, M. Stojanovic, and M. Médard, "Random linear network coding for time division duplexing: When to stop talking and start listening," *CoRR*, vol. abs/0809.2350, 2008, informal publication. [Online]. Available: <http://dblp.uni-trier.de/db/journals/corr/corr0809.html#abs-0809-2350>
- [11] C.-C. Chao, C.-C. Chou, and H.-Y. Wei, "Pseudo random network coding design for IEEE 802.16m enhanced multicast and broadcast service," in *Vehicular Technology Conference (VTC 2010-Spring)*, May 2010, pp. 1–5.
- [12] Z. Liu, C. Wu, B. Li, and S. Zhao, "Uusee: Large-scale operational on-demand streaming with random network coding," in *IEEE International Conference on Computer Communications*, 2010, pp. 2070–2078.
- [13] M. V. Pedersen, J. Heide, and F. H. Fitzek. (2011, Feb.) The cone project homepage. [Online]. Available: <http://mobdevtrac.es.aau.dk/cone/wiki/CodeOverhead>