Selecting Optimal Parameters of Random Linear Network Coding for Wireless Sensor Networks

Heide, Janus; Zhang, Qi; Fitzek, Frank

Published in:
Vehicular Technology Conference (VTC Fall), 2013 IEEE 78th

DOI (link to publication from Publisher):
10.1109/VTCFall.2013.6692403

Publication date:
2013

Document Version
Early version, also known as pre-print

Selecting Optimal Parameters of Random Linear Network Coding for Wireless Sensor Networks

Janus Heide*, Qi Zhang†, Frank H.P. Fitzek*,
*Department of Electronic Systems, Faculty of Engineering and Science, Aalborg University, Denmark
†Aarhus University, Denmark
Email: jah@es.aau.dk, qz@iha.dk, ff@es.aau.dk

***Abstract*—This work studies how to select optimal code parameters of Random Linear Network Coding (RLNC) in Wireless Sensor Networks (WSNs). With Rateless Deluge [?] the authors proposed to apply Network Coding (NC) for Over-the-Air Programming (OAP) in WSNs, and demonstrated that with NC a significant reduction in the number of transmitted packets can be achieved. However, NC introduces additional computations and potentially a non-negligible transmission overhead, both of which depend on the chosen coding parameters. Therefore it is necessary to consider the trade-off that these coding parameters present in order to obtain the lowest energy consumption per transmitted bit. This problem is analyzed and suitable coding parameters are determined for the popular Tmote Sky platform. Compared to the use of traditional RLNC, these parameters enable a reduction in the energy spent per bit which grows as the generation size grows. These results also indicate that the use of high field sizes could be problematic from an energy point of view due to the additional complexity.**

I. Introduction

In WSNs it is necessary to upgrade programs on the sensor nodes after deployment, e.g. to fix bugs, replace program modules or tune module parameters [?], which is referred to as code dissemination. Usually it is impractical or impossible to manually reprogram deployed sensor nodes. Therefore, the update must be disseminated over the air, referred to as OAP. OAP is crucial for the success of WSN as it facilitates management, maintenance, and adaptive WSN applications [?]. The challenges of OAP for WSNs are due to the extreme resource constraints of the sensor nodes in terms of energy capacity, memory size, and computation capability. Additionally, OAP requires 100% reliability over the unreliable wireless links.

Deluge [?] is a widely used OAP protocol which aims to achieve reliable code dissemination with low control message overhead by dynamic advertisement, retransmission request suppression and achieves rapid code dissemination through exploiting spatial multiplexing. Subsequently rateless Deluge has been proposed [?] which applied RLNC [?] to Deluge, and demonstrated a reduction in the number of transmitted data packets by 15-30% and control packets by 50-80%. With RLNC a single linear combination can fix different missing packets at different receivers. Thus the feedback from a receiver is reduced to the number of missing packets. By reducing the number of transmitted data and control packets, rateless Deluge offers the prospect of significant energy savings. However, the added memory and computational overhead due to the coding processes remains uninvestigated so far.

RLNC provides a theoretically efficient method for coding [?], as the overhead can asymptotically approach zero. In practical applications of RLNC, the original data is typically divided into generations [?]. This ensures that the performance of RLNC is independent of the data size and reduces both the computational work and the decoding delay. Unfortunately, it also increases the probability of receiving linearly dependent symbols and introduces the need for additional signaling [?]. The probability of linear dependency and the coding vector which describes the performed coding, adds to the transmission overhead. Additionally, the coding processes introduces computational complexity in terms of memory access and computations. Both the overhead and the computational complexity adds to the total energy spent to communicate a bit and depends on coding parameters such as generation size, field size and density. The generation size defines the decoding delay and thus cannot be chosen freely, but depends on the content that is distributed. It has also been shown that a high field size and density does not always result in the lowest overhead [?]. To achieve the maximal energy savings, the parameters of the used RLNC code must be carefully selected to obtain the best trade-off between complexity and transmission overhead for the considered platform and scenario.

In this work the deployment of practical RLNC for sensor nodes is investigated. The primary contribution is the analysis of the computational complexity of RLNC in terms of memory access and symbol additions, and the resulting transmission overhead in terms of retransmissions and coding vector representation. The expressions are used to determine coding parameters that minimize the energy for the Tmote platform under consideration. The numerical results show that the achieved reduction in energy used per bit grows with the generation size. Thus using optimal coding parameters enables the use of higher generation sizes on energy constrained platforms. This could improve the performance of the protocols which utilize NC to achieve reliability.

The remainder of this paper is structured as follows. In Section II the cost due to coding is analyzed in terms of overhead due to generation size, field size, density, and the coding vector, and the computational complexity in terms of memory access and row operations. Based on this the energy used per transmitted bit for the source and receiver is analyzed. Numerical results are presented in Section III, discussed in Section IV and final conclusions are drawn in Section V.

II. Applying RLNC in WSNs

Coding at all nodes is performed using RLNC as introduced in [**?**]. Here follows a brief introduction, we refer interested readers to [**?**], [**?**] for more thorough introductions. The data to be transmitted from a source $\boldsymbol{B}$ is divided into generations $\boldsymbol{M}$. Each generation constitutes g symbols, where each symbol has the size m bits and is represented by $\lceil \frac{m}{\log_2(q)} \rceil$ field elements in a finite field $\mathbf{F}_q$ of size q. Each generation is represented by $g \cdot m$ bits.

During encoding, a coding vector $\boldsymbol{v}$ is generated comprising elements in $\mathbf{F}_q$. A coded symbol $\boldsymbol{x}$ is created by multiplying the coding vector with the original data $\boldsymbol{x} = \boldsymbol{M} \cdot \boldsymbol{v}$. The density d of the coding vector is defined as the ratio of non-zero elements in the coding vector, i.e., $P(\boldsymbol{v}_i \neq 0) = d, 0 < d \leq 1 - \frac{1}{q}$.

At a receiver, the symbols and coding vector pairs can be decoded as $\hat{\boldsymbol{M}} = \hat{\boldsymbol{X}} \cdot \hat{\boldsymbol{V}}^{-1}$ when g linearly independent symbols are collected. To collect g linearly independent symbols, on average $g + \rho$ symbols must be collected where ρ depends on the used code. A receiver may recode a symbol $\tilde{\boldsymbol{x}}$ and coding vector $\tilde{\boldsymbol{v}}$ based on the received symbols and coding vectors arranged in $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{V}}$, by creating a local recoding vector $\boldsymbol{h}$, $\tilde{\boldsymbol{x}} = \hat{\boldsymbol{X}} \cdot \boldsymbol{h}$, $\tilde{\boldsymbol{v}} = \hat{\boldsymbol{V}} \cdot \boldsymbol{h}$. Notations used in the analysis are listed in Table I.

TABLE I: Notation

Symbol	Unit	Definition
g	symbols	Generation size
q	-	Field size
d	-	Density, ratio of elements that are non-zero
δ	-	Ratio of elements in $\boldsymbol{v}$ that are drawn at random
m	bits	Symbol size
b	-	Element in $\mathbf{F}_2$
a, c, e	-	Elements in $\mathbf{F}_q$
$\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$	-	Symbols, vectors of elements in $\mathbf{F}_q$
$\boldsymbol{v}$, $\boldsymbol{u}$		Coding vector
α	-	Expected number of retransmissions per generation due to the field size
β	-	Expected number of retransmissions per generation due to the density
γ	bits	Size of a coding vector size (compressed)
μ	bits	Size of a coding vector size (uncompressed)
r	-	Rank at a receiver
i, j, k		Counter variables
ρ	-	Expected number of redundant symbols
σ	bits	Overhead
τ	bits	Expected total bits per generation
R	kb/s	data rate
ϵ	-	Packet erasure probability
ϱ	bits	microprocessor operation length
$\dot{\oplus}$	-	Row multiplication-addition
RD	bits	Number of bits read from memory
WR	bits	Amount of bits written into memory

The size of the coding vector in memory is defined as

$$\mu = g \cdot \log_2(q)$$

As the elements in the coding vector are drawn at random from $\mathbf{F}_q$, the density d can be calculated as follows.

$$d = d(\boldsymbol{v}) = P(\boldsymbol{v}_i \neq 0) = \delta \cdot (1 - q^{-1})$$

Where $\delta > 0$ denotes the ratio of elements drawn at random from $\mathbf{F}_q$, hence $0 < d < \delta \leq 1$. For high q, $d \approx \delta$. When q is small, the number of zero elements drawn at random becomes significant. The following constraint ensures that the probability of generating the zero vector remains low when each element is drawn at random.

$$g \cdot d > 1 \leftrightarrow \delta > \frac{1}{g \cdot (1 - q^{-1})}$$

A. Overhead

When RLNC is applied two types of overhead are introduced in terms of extra bits that must be transmitted. One source of overhead is due to linear dependency. If a received symbol is a linear combination of the already received symbols, then the newly received symbol does not provide any new information to the receiver. The other source of overhead is the coding vector that has to accompany each coded symbol to communicate the operations performed during the coding procedure.

The overhead depends on the coding parameters such as the generation size, field size and density. Here we will briefly introduce the necessary expressions, for details we refer to [**?**].

1) Generation Size: The generation size g defines the number of symbols over which coding is performed. A higher g reduces the probability of linearly dependent symbols, but at the same time increases the delay and the complexity at the receiver, which also increase the energy consumption. Hence g cannot be chosen freely, but is a trade-off between delay, decoding complexity, and linear dependency.

2) Field Size: The field size q defines the size of $\mathbf{F}_q$ which represents the elements in symbols and coding vectors. Small values of q impose lower computational complexity from a theoretical point of view, but typically g is chosen so that it fits with a native data container to enable efficient implementations, e.g. $g = \{2, 2^8, 2^{16}\}$.

The rank at the receiver is denoted r, hence $g - r$ pivot elements have not been identified. For an incoming coded symbol to be linearly dependent, the corresponding $g - r$ elements must be zero. An element drawn at random is zero with probability $\frac{1}{q}$, hence the probability that all elements are zero is given by $(1/q)^{g-r}$. Hence, the probability that the symbol contains a novel pivot element can be found and the expected number of transmissions can be calculated. The expected number of retransmissions of symbols in one generation can be calculated by summing the expected number of retransmissions at all possible ranks of the receiver.

$$\begin{aligned} \alpha(q, g) &= \sum_{r=0}^{g-1} \left(\left(1 - \frac{1}{q^{g-r}} \right)^{-1} - 1 \right) \\ &= \sum_{r=0}^{g-1} \left(\frac{1}{q^{g-r} - 1} \right) \end{aligned} \tag{1}$$

3) Density: The density d defines the number of symbols that are combined to create a coded symbol. Reducing d can reduce the computational load, but can also increase the probability of linear dependency during decoding.

The number of symbols received by the receiver is denoted k. Necessary but insufficient conditions for the receiver to attain full rank are; $k \geq g$, and that all pivot positions are non-zero in at least one of the received coding vectors. The probability that a pivot is zero for all k received coding vectors is $(1-d)^k$. The probability that for all g pivot positions, at least one of the k coding vectors is non-zero is given by $(1-(1-d)^k)^g$.

To find the expected number of symbols that must be received in addition to g we sum the probability that all k received symbols are zero for (at least) one pivot position.

$$\beta(d,g) = \sum_{k=g}^{\infty} \left(1 - \left(1-(1-d)^k\right)^g\right) \quad \text{where } 0 < d \leq 1 - q^{-1} \tag{2}$$

4) Coding Vector: The coding vector communicates all operations performed on the symbol during encoding and recoding. Such a coding vector can be efficiently represented by a bit array and a set of scalars [**?**]. The bit array $\boldsymbol{b} = [b_1, b_2 \ldots, b_g]$ indicates the non-zero elements. The scalars, $[a, c, \ldots, e]$, define the values of the non-zero elements.

b_1	b_2	$\ldots$	b_g	a	c	$\ldots$	e

The bit array can be represented by g bits. If the bit array is compressed with an optimal code, the necessary amount of bits can be reduced to the entropy of the bit vector, $H(\boldsymbol{b})$, which can be calculated from d and g. Each of the scalars can be represented by $\log_2(q)$ bits, on average there are $g \cdot d$ such scalars for each coded symbol.

$$\gamma = H(\boldsymbol{b}) + \log_2(q) \cdot g \cdot d \tag{3}$$

When symbols are transmitted over the network, the coding vector is compressed to γ bits. However, it is represented in its uncompressed form of μ bits when symbols are processed at the source and the receivers . This is to ensure fast addition of two vectors.

5) Total: From these expressions, the following metrics can be expressed. The expected number of redundant symbols, $\rho \geq 0$. The overhead, $\sigma \geq g \cdot \gamma$. The expected total bits per generation $\tau \geq g \cdot m$.

$$\rho = (\alpha + \beta) \tag{4}$$

$$\sigma = (\alpha + \beta) \cdot m + (g + \alpha + \beta) \cdot \gamma \tag{5}$$

$$\tau = (g + \alpha + \beta) \cdot (m + \gamma) \tag{6}$$

B. Computational Complexity

The complexity for a generation is analyzed using the metric of symbol multiplication-addition, which we denote $\dot{\oplus}$. A symbol multiplication-addition means that the elements in a symbol are multiplied with a scalar and added element-wise to the elements in another symbol, $\boldsymbol{x} = c \cdot \boldsymbol{y} \oplus \boldsymbol{z}$, where $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ are vectors of elements in $\mathbf{F}_q$ and $c \in \mathbf{F}_q$.

In the case of encoding, the coding vector $\boldsymbol{v}$ is read one element at a time and the appropriate operations are performed. However, during decoding an additional operation is performed $\boldsymbol{v} = c \cdot \boldsymbol{u} \oplus \boldsymbol{v}$, in order to update the coding vector as decoding is performed.

Note that for the binary field $\mathbf{F}_2/\{0\} = \{1\}$, and therefore each symbol operation degenerates to the element row addition, which in $\mathbf{F}_2$ is the elementwise XOR.

Uniform density for all symbols is assumed, hence $d = E\left(d(v_i)\right) \forall i$, where v_i is an element in a coding vector.

1) Encoding: The expected number of encoded symbols is $(g + \rho)$. To encode a single symbol, on average $g \cdot d$ symbols are combined.

$$\dot{\oplus}_{\text{enc}} = (g + \rho) \cdot g \cdot d \tag{7}$$

2) Decoding: During decoding we assume a brute force decoding algorithm that attempts to identify an unseen pivot element by subtracting previously received and partially decoded symbols from it. Thus no sorting or swapping to reduce the amount of *fill-in* is utilized. When a symbol is received, its first non-zero element is identified which is its pivot element, and the symbol is normalized with respect to the pivot[1]. If none of the previously received and partially decoded symbols has the same pivot as the newly received symbol, then the element is decoded and stored. If the pivot element was identified in a previously received symbol, then the previously received symbol is subtracted from the new symbol, a new pivot element is identified, and the process is repeated.

We note that linearly dependent symbols can be detected and discarded before decoding them by first performing row operations on the coding vector only.

When two coding vectors $\boldsymbol{y}$ and $\boldsymbol{z}$ are added, the probability that an element in the resulting vector is the zero element, $P\left(\boldsymbol{y}_i \oplus \boldsymbol{z}_i = 0 | d\right)$, where $d = d(\boldsymbol{y}) = d(\boldsymbol{z})$, is given by the probability that both elements in $\boldsymbol{y}$ and $\boldsymbol{z}$ are zero, and the probability that both are the same non-zero value. The probability that the combination of two scalars over $\mathbf{F}_q$ equals zero is the probability that they are both zero plus the probability that they both have the same non-zero value, see Equation (8).

$$P\left(\boldsymbol{y}_i \oplus \boldsymbol{z}_i = 0 |\, d\right) = (1-d)^2 + d^2 \cdot \frac{1}{q-1} \tag{8}$$

For each row the symbol has a non-zero probability of being non-zero in which case it will be combined with the existing row, which can increase its density. If the element corresponding to the pivot position is zero, with probability $(1-d)$, it is inserted and the density of the symbol remains the same. With probability d_j the corresponding pivot element is non-zero in the received symbol in which case the density of the remaining $g-j$ elements in the symbol is $P(\boldsymbol{y}_i \oplus \boldsymbol{z}_i \neq 0 | d_j)$. As rows are inserted and decoded their density will increase as more and more rows are inserted. The density of the partially decoded rows is recursively defined in Equation (9). With probability $(1-d)$ the density remain the same, in the case where the

[1]All elements in the symbol are multiplied with the inverse of the pivot element.

scalar was zero, with probability d the density increases, if the scalar was non-zero.

$$d_{j+1} = (1 - d_j) \cdot d_j + d_j \cdot P(\boldsymbol{y}_j \oplus \boldsymbol{z}_j \neq 0 | d_j) \qquad (9)$$
$$d_0 = d$$

To calculate the accumulated complexity for all the elements that must subsequently be removed sum over all the resulting coding vectors, excluding their pivot element. The first i scalars in each row are zeros, where i is the row index, the pivot element is 1 and the remaining elements are non-zero with probability d_j. Hence the expected number of non-zero elements can be calculated with Equation (10).

$$\dot{\oplus}_{\text{dec}} = \sum_{j=1}^{g-1} (g - j - 1) \cdot d_j = \frac{g \cdot (g-1)}{2} \cdot d_j \qquad (10)$$

This only accounts for the row operations performed during forward substitution. Backwards substitution is performed in a similar way, only upwards, which can be expressed as $\sum_{j=g-1}^{1} \left(\sum_{i=j-1}^{0} d_i \right) = \sum_{j=g-1}^{1} (g - j - 1) \cdot d_j = \dot{\oplus}_{\text{dec}}$ and hence the number of operations during backwards substitution is the same as during forward substitution. A minor difference is that during backwards substitution the operations on the coding vectors can be omitted [**?**].

C. Memory Access

We assume that the algorithm reads and writes symbols and coding vectors on the full representation of μ bits.

1) Encoding: The expected number of symbols which are combined to create the $g + \rho$ symbols is given by $\dot{\oplus}_{\text{enc}}$, hence $\dot{\oplus}_{\text{enc}}$ symbols are read and $g + \rho$ symbols and their coding vectors are written. The amount of bits read and written during encoding RD_{enc} and WR_{enc}, respectively, can be expressed as.

$$\text{RD}_{\text{enc}} = \dot{\oplus}_{\text{enc}} \cdot m \qquad (11)$$
$$\text{WR}_{\text{enc}} = (g + \rho) \cdot (m + \mu) \qquad (12)$$

2) Decoding: During forward substitution g symbols and coding vectors are read. The ρ symbols that are linearly dependent can be detected via their coding vector, and thus it is only necessary to read the coding vector of these ρ symbols. The $\dot{\oplus}_{\text{dec}}$ symbols and their coding vectors used to reduce the incoming symbols are read. Finally the g resulting symbols and coding vectors are written.

During backward substitution all g symbols and coding vectors are read one at a time. Additionally the $\dot{\oplus}_{\text{dec}}$ symbols and coding vectors used to decode the g symbols are read. Finally the g fully decoded symbols are written, we note that it is not necessary to write the coding vectors.

The amount of bits read and written during decoding RD_{dec} and WR_{dec}, respectively, can be expressed as.

$$\text{RD}_{\text{dec}} = 2 \cdot (g + \dot{\oplus}_{\text{dec}}) \cdot (m + \mu) + \rho \cdot \mu \qquad (13)$$
$$\text{WR}_{\text{dec}} = g \cdot (2 \cdot m + \mu) \qquad (14)$$

D. Energy

We consider the following components that contribute to energy usage at a source and receivers: the network interface, the processing unit, and the memory.

The energy used for memory access and XOR'ing is obtained from [**?**], where the authors perform a series of measurements on the Tmote Sky based on the MSP430 microprocessor and the CC2420 radio chip. The reported results can be verified with a few simple calculations. Based on the reported setup, the supply voltage at the chip was $\frac{1500\Omega}{1500\Omega + 101\Omega} \cdot 2.7\text{V} = 2.53\text{V}$ since it was in series with a resistor. The energy used for one NOP operation can be calculated based on the specified current and operating frequency from the data sheet $1.8\text{mA} \cdot 2.53\text{V} / 10^6\text{Hz} = 4.55\text{nJ}$. This value is close to the reported value of 4.4nJ for a NOP operation.

The expected energy consumed for sending and receiving a single bit can be calculated using Equation (15).

$$E_{\text{tx/rx}} = \frac{O_{\text{phy}} + \gamma + m}{m} \cdot \frac{U \cdot I}{R} \qquad (15)$$

Where O_{phy} is the packet overhead added on the physical layers, U is the voltage drop over the radio chip, and I is the current drawn by the chip. The necessary values can be obtained from the data sheet for the used C2420 chip [**?**].

TABLE II: The used energy values in the evaluation.

Symbol	Value	Description
E_{xor}	4.8 nJ	Energy XOR'ing two 16 bit words
E_{rd}	13.2 nJ	Energy reading 16 bits from memory
E_{wr}	17.0 nJ	Energy writing 16 bits to memory
E_{rx}	155.1 nJ	Energy receiving a bit
E_{tx}	143.5 nJ	Energy sending a bit

Additionally, in the considered scenario the following parameters are defined, the generation size $g \in [8, 128]$, the field size $q = 2$, the symbol size $m = 880$ bits, the data rate $R = 250$ kbs, the CPU instruction length $\varrho = 16$ bits, and the packet erasure probability $\epsilon = 0.07$ [**?**].

Finally we can define the energy consumed per transmitted generation from the application layer at a sender.

$$E_{\text{sender}} = \frac{\text{RD}_{\text{enc}}}{\varrho} \cdot E_{\text{rd}} + \frac{\text{WR}_{\text{enc}}}{\varrho} \cdot E_{\text{wr}} + \frac{\dot{\oplus}_{\text{enc}} \cdot m}{\varrho} \cdot E_{\text{xor}} + \frac{1}{1 - \epsilon} \cdot \tau \cdot E_{\text{tx}} + O_{\text{ack}} \cdot E_{\text{rx}} \qquad (16)$$

As all transmitted packets are processed on the physical layer at the receiver, $\frac{1}{1-\epsilon}$ is included in the term $\frac{1}{1-\epsilon} \cdot \tau \cdot E_{\text{rx}}$. Thus the energy per received generation in the application layer is defined as.

$$E_{\text{receiver}} = \frac{\text{RD}_{\text{dec}}}{\varrho} \cdot E_{\text{rd}} + \frac{\text{WR}_{\text{dec}}}{\varrho} \cdot E_{\text{wr}} + \frac{1}{1 - \epsilon} \cdot \tau \cdot E_{\text{rx}} + \frac{2 \cdot \dot{\oplus}_{\text{dec}} \cdot (m + \mu)}{\varrho} \cdot E_{\text{xor}} + O_{\text{ack}} \cdot E_{\text{tx}} \qquad (17)$$

The energy per application-layer bit is thus $\frac{E_{\text{sender}}}{g \cdot m}$ and $\frac{E_{\text{receiver}}}{g \cdot m}$, respectively. Note that only one acknowledgment per generation is needed, instead of one for each packet.

III. Results

We consider the case of a single source and a single receiver that communicate over an erasure channel with erasure probability $\epsilon = 0.07$ [**?**].

To obtain results, minimization of several expressions is performed wrt. d, subject to $1 \geq \delta > \frac{1}{g \cdot (1-q^{-1})}$ and $q = 2$. Thus d is varied in order to minimize the energy spent. The expression $\min(\frac{E_{\text{sender}}}{g \cdot m})$ minimizes at the sender, and $\min(\frac{E_{\text{receiver}}}{g \cdot m})$ minimizes at the receiver. $\min(\frac{E_{\text{sender}}+E_{\text{receiver}}}{2 \cdot g \cdot m})$ minimizes the average energy usage, or alternatively the total/system usage.

In some cases it might also be relevant to minimize $\min(\max(\frac{E_{\text{sender}}}{g \cdot m}, \frac{E_{\text{receiver}}}{g \cdot m}))$ in order to maximize the system operation time, if it is assumed that the sender and the receiver have the same energy available. However, for the case considered here the energy spent at the receiver was always higher than that at the sender, and is therefore equal to $\min(\frac{E_{\text{receiver}}}{g \cdot m})$.

Fig. 1 shows the resulting densities that minimize energy for the source and receiver independently and jointly. The x-axis denotes the generation size, and the y-axis denotes the optimal density.

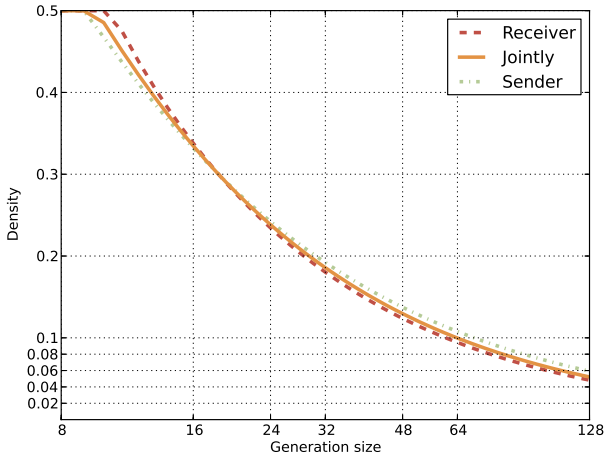

Fig. 1: The resulting density for the minimizing energy consumed on the sender, the receiver, and jointly.

First, it is apparent that as g grows, the optimal density becomes much smaller than $1 - q^{-1}$. Thus a high density assumed in most works is very far from the optimal solution. Secondly, the resulting densities that minimize the energy spent per bit are very similar for the sender and receiver. It is somewhat surprising that the densities are almost the same, but fortunately it means that optimizing for e.g. the receiver still provides decent results at the sender.

Fig. 2 shows the energy per bit at different values of g for the two cases; traditional RLNC where $d = 1 - q^{-1}$, and the optimal density for each generation size. Results are provided for the source and receiver independently, and jointly for a system comprising one source and one receiver. The x-axis denotes the generation size, and the y-axis denotes the expected energy spent sending and/or receiving one bit.

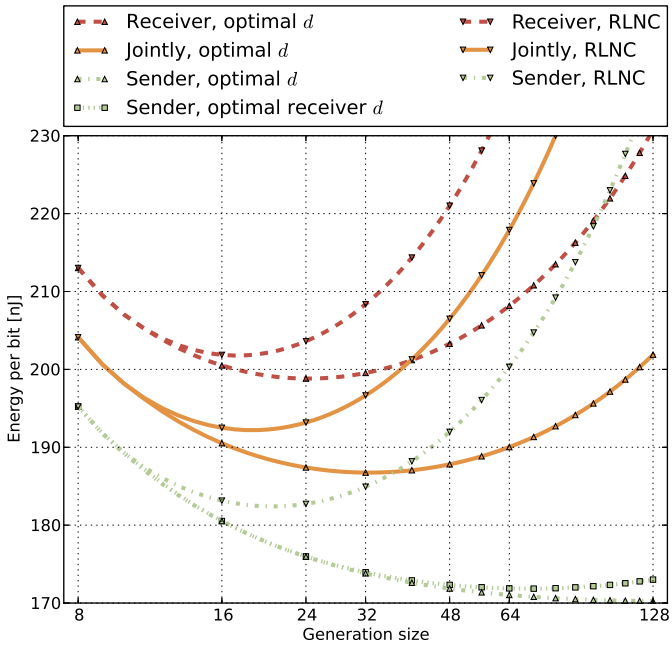

Fig. 2: The energy used per transmitted bit for the sender, receiver, and jointly, for the two cases, RLNC and when the energy per bit is minimized wrt. to d.

Compared to RLNC, using the optimal density results in a lower or identical energy consumption per bit for all values of g. The difference between the two approaches goes to zero as g goes towards one, which is unsurprising as the optimal densities obtained in Fig. 1 show that in this case the optimal density approaches $d = 1 - q^{-1}$. Conversely, as g increases the optimal density goes towards zero.

For the case of optimal density, the lowest energy per bit at the sender is obtained when g is chosen as large as possible. To minimize energy at the receiver, the best trade-off for g is in the range between 16 and 32. This will also maximize the network life time, if both the sender and receiver have same energy available. From the perspective of the entire system, g should be chosen in the range between 24 and 48. It is worth noticing that for RLNC a good choice of g is slightly different as it should be between 12 and 24.

The receiver achieves the lowest energy per bit at $g = 17$ for RLNC, and $g = 24$ for the optimal density. However, in general g is not chosen to minimize the energy per bit, but instead based on the delay requirements and/or the network topology. At the highest $g = 48$ considered in [**?**], RLNC uses 8.7% more energy per bit compared to when the optimal density is used. For the energy per bit obtained with RLNC and $g = 20$ [**?**], the same energy per bit is obtained at $g = 44$ when the optimal density is used.

If the optimized density for the receiver is used at the sender, the cost in terms of energy is only slightly increased at the sender, and only when g increases above 64. This is worth noticing because the receiver incurs the highest energy cost, and often the number of receivers will be higher than the number of senders.

IV. Discussion

The obtained results can be put into context by drawing parallels to the interesting work on rateless Deluge [**?**]. The authors use a field size of 2^8 implemented with a lookup table. This field size is often used in studies of RLNC as it provides a decoding probability that can be ignored for all practical considerations, and its implementation is straightforward. With a lookup table, field operations are performed by reading from memory which is relatively cheap at 26.4 nJ for two bytes, but it is still five times more expensive than an XOR at 4.8 nJ, which implements addition in $\mathbf{F}_2$. If a multi-hop scenario is considered, the important recoding of NC feature should also be employed. However, when recoding it is difficult, possibly impossible, to utilize the neat trick of a seed combined with a pseudo-random generator [**?**]. Until a suitable solution to this problem is presented, RLNC with a *high* field size and density will result in a coding vector overhead of $g \cdot \log_2(q)$ bits for each coded symbol, which is particularly problematic for WSN where the packet size is small.

The presented results also provide insights into scenarios with multiple sources and/or multiple sinks. For multiple receivers with homogeneous packet erasure probability, the system energy consumed per bit will approach $\min(\frac{E_{\text{receiver}}}{g \cdot m})$, because when the number of receivers grows, the energy spent by the sender will become negligible. For multiple receivers with different erasure probabilities, it can simply be assumed that $\epsilon = \max(\boldsymbol{\epsilon})$, where $\boldsymbol{\epsilon}$ is the vector of erasure probabilities from the sender to the individual receivers. This is because the source(s) must only overcome the maximum erasure probability, ϵ, in order to successfully transmit data to all the receivers. Namely, if the receiver with the highest erasure probability is satisfied, then so are the remaining receivers. With multiple senders, the number of transmissions can simply be spread among the senders. In such a case the receiver(s) would still be the node(s) which consume the highest energy per bit.

As sensor nodes evolve in the future with more powerful hardware, more memory and higher computational capabilities, the ratio between transmission cost and computational cost may change. Additionally, the evolution of new use cases for WSNs may also impact this ratio since generally the cost per transmitted bit increases with the communication range and decreases as the transmission rate increases. If the cost of computations decreases relatively to the transmission cost, it should be expected that higher values of q, g, and d will provide a lower energy per bit cost. For higher field sizes, an interesting future direction would be the use of iterative algorithms which would remove the need for a lookup table. Additionally, it could enable the use of multiple fields simultaneously which would permit adaptation of q to different scenarios. Another promising direction is the development of code variants that enables faster decoding, and faster decoding algorithms. Such approaches should be designed jointly to permit the decoding algorithms to exploit special structures in the coded symbols.

V. Conclusion

In this work, we have analyzed the performance of RLNC on sensor nodes in terms of transmission overhead and computational complexity, both as a function of typical adjustable coding parameters. Expressions for the energy per bit were found and used to evaluate the energy cost based on values obtained from the Tmote Sky platform.

The numerical results showed that for the binary field, the lowest energy cost at the receiver is obtained for a generation size in the range $[16, 32]$ and a density in the range $[0.2, 0.3]$. When the sender and the receiver are considered jointly a generation size in the range $[24, 48]$ and a density in the range $[0.1, 0.2]$ should be used. This is significantly different from most existing studies where the generation size, field size, and density are typically assumed as high as possible, and the overhead in terms of transmissions and computational complexity is ignored. However, both of these overheads add to the energy consumption of the communication system and are therefore particularly relevant for WSNs.

Two interesting questions are left for future work. The impact on the overlaying protocols, where lower generation sizes generally require more feedback, and the extension of the analysis to higher fields.

References

[1] A. Hagedorn, D. Starobinski, and A. Trachtenberg, "Rateless deluge: Over-the-air programming of wireless sensor networks using random linear codes," in *Proceedings of the 7th international conference on Information processing in sensor networks*, ser. IPSN '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 457–466.

[2] Q. Wang, Y. Zhu, and L. Cheng, "Reprogramming wireless sensor networks: challenges and approaches," *Network, IEEE*, vol. 20, no. 3, pp. 48 – 55, may-june 2006.

[3] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, ser. SenSys '04. New York, NY, USA: ACM, 2004, pp. 81–94.

[4] 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.

[5] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," *Proceedings of the annual Allerton conference on communication control and computing*, vol. 4, pp. 40–49, 2003.

[6] P. Maymounkov, N. J. A. Harvey, and D. S. Lun, "Methods for Efficient Network Coding," *44th Allerton Annual Conference*, 2006.

[7] J. Heide, M. V. Pedersen, F. H. Fitzek, and M. Médard, "On code parameters and coding vector representation for practical rlnc," in *IEEE International Conference on Communications (ICC) - Communication Theory Symposium*, Kyoto, Japan, jun 2011.

[8] C. Fragouli, J. Boudec, and J. Widmer, "Network coding: an instant primer," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 63–68, 2006.

[9] J. Heide, M. V. Pedersen, F. H. Fitzek, and T. Larsen, "Cautious view on network coding - from theory to practice," *Journal of Communications and Networks (JCN)*, vol. 10, no. 4, pp. 403–411, December 2008.

[10] J. Heide, M. V. Pedersen, and F. H. Fitzek, "Decoding algorithms for random linear network codes," in *IFIP International Conferences on Networking - Workshop on Network Coding Applications and Protocols (NC-Pro)*, ser. Lecture Notes in Computer Science, vol. 6827, Valencia, Spain, may 2011, pp. 129–137.

[11] N. Lane and A. Campbell, "The influence of microprocessor instructions on the energy consumption of wireless sensor networks," in *Third Workshop on Embedded Networked Sensors (EmNets)*, vol. 34, 2006.

[12] "Chipcon CC2420 Datasheet: http://focus.ti.com/lit/ds/symlink/cc2420.pdf, Texas Instruments," 2007.