Revolving Codes: High Performance and Low Overhead Network Coding

Niloofar Yazdani, and Daniel E. Lucani
DIGIT and Department of Engineering, Aarhus University, Denmark
{n.yazdani, daniel.lucani}@eng.au.dk

***Abstract*—Revolving Codes (ReC) are introduced as an alternative to other network codes to reduce per packet and total overhead, and reduce the probability of linearly dependent coded packets. Furthermore, Revolving Codes reduce the costs to intermediate nodes by introducing a recoding scheme based on XOR operations. Revolving Codes are well suited for new applications transmitting small payloads, e.g., IoT, Industry 4.0. Our numerical results show that ReC outperforms RLNC in total overhead by as much as two orders of magnitude and it outperforms Fulcrum codes by as much as two orders of magnitude in terms of the overhead caused by linearly dependent packets.**

***Index Terms*—Network coding, RLNC, overhead reduction.**

I. Introduction

Network coding (NC) can help to improve a network's throughput, reliability, and scalability [1]. These gains are a result of NC's ability to recombine coded packets at intermediate nodes without the need to decode them. Recoding introduces flexibility in the presence of packet losses by introducing extra redundancy in lossy links [1]. Random Linear Network Coding (RLNC) [2] provides a practical approach to NC, where the source and intermediate nodes can generate coded packets by randomly combining original packets or already coded packets over finite fields. However, RLNC can introduce considerable overhead to signal how a coded packet was constructed as well as overhead from linearly dependent packets, i.e., additional transmissions due to the fact that a coded packet was not useful for decoding the original data. Several NC approaches have been proposed to address this issue. One alternative is to use a seed value for a pseudo-random number generator which can significantly reduce the overhead, however, no effective recoding approach is available for this strategy. Another alternative is to use sparse coding techniques. For example, coding schemes that use a small set of allowed coefficients [3], but with more complex recoding algorithms; based on band matrices [4], which require large buffers to save different subsets; or window-based network codes such as Perpetual codes [5] and caterpillar RLNC [6]. Sparse techniques typically required carefully designed recoding strategies to maintain the structure of the sparse code. Another set of schemes focuses on exploiting finite field characteristics and code concatenation, e.g., Fulcrum codes [7], or developing new finite field constructions as the basis for new codes, e.g., Telescopic codes [8]. These schemes typically require careful tuning and a trade-off between dependency overhead and signaling overhead, but they enjoy simple recoding mechanisms. Despite the improvements from the latter, there is still a need for simultaneously reducing dependency and signaling overhead and maintaining simpler encoders and/or recoders.

In this paper, the following contributions are made to address the aforementioned problems:

- Revolving codes (ReC) are proposed not only to reduce the overall overhead considerably, but also to reduce the probability of linearly dependent packets, while maintaining a simple encoder/decoder.
- A simple recoding scheme that is less computationally demanding and that preserves ReC's structure.

Our results show that our scheme's total overhead is comparable to that of fine-tuned Fulcrum codes, while delivering a much lower probability of linearly dependent packets (equivalent to RLNC) resulting in two orders of magnitude lower overhead from linearly dependent packets. This means that ReC can deliver RLNC performance while maintaining the low overhead of Fulcrum codes.

II. Description of the scheme

A. Idea

Small finite fields introduce less overhead per packet caused by coefficients and less computational complexity in the context of NC, but they typically result in a higher probability of linearly dependent packets [1]. Large finite fields decrease the probability of having dependent packets at the expense of more overhead due to vectors of coefficients. The idea behind ReC is to apply large finite fields, but operate them in part as small ones, both from a perspective of signaling and recoding. The value of coefficients for each packet revolves around a common value which is drawn from a large finite field by applying bit flips to a subset of the common value. The specific bits to be flipped is deterministic but changes depending on the corresponding packet's order within the group of packets (generation) being combined. The specific bits changed per coefficient with respect to the common value are fewer than the finite field size, thus decreasing overhead.

B. Design

Definition 1: An *ReC(q,b,t)* code is a revolving code using a finite field $\mathbb{F}_q$, flips at most b bits in each coefficient around a common value which is chosen uniformly at random from $\mathbb{F}_q$, and transmits t bits per coefficient aside from the $log_2(q)$ bits from the common value.

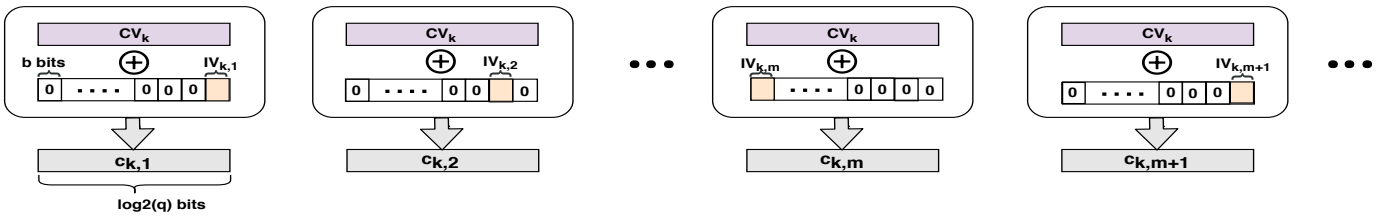

Fig. 1: The approach of making the coefficients.

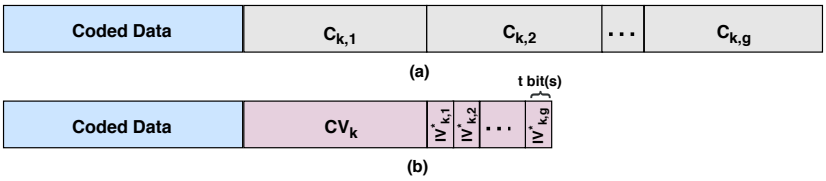

Fig. 2: Packet structures for a) RLNC, and b) Revolving codes.

In the following, the operations at the encoder, recoder and the decoder are described.

1) Encoder: To generate the kth coded packet, X_k, the original g (generation size) packets, P_1, P_2, ..., P_g are linearly combined by multiplying them by coefficients $c_{k,i}$, assuming the data in each packet is a concatenation of symbols in a Finite Field of size q ($\mathbb{F}_q$):

$$X_k = \sum_{i=1}^{g} c_{k,i} P_i, \tag{1}$$

where coding coefficients, $c_{k,i}$s, for coded packet X_k, are a combination of an element drawn at random from $\mathbb{F}_q$ called *common value (CV_k)*, and random values called *individual values ($IV_{k,i}$)* of length b bits, where b is normally equal to 1 or 2. To make the coefficient, $c_{k,i}$, the CV_k is added by $IV_{k,i}$ which is shifted, circularly in a register of $\log_2(q)$ bits, $i-1$ times by b bits. Fig. 1 depicts the process of making the coefficients for each coded packet using a bit-by-bit XOR operation as addition.

Each coded packet has a header that contains the information associated with coding coefficients. Each coefficient is of length $\log_2(q)$ bits. However, instead of having a header of length $g \cdot \log_2(q)$ bits, each coding coefficient, $c_{k,i}$, can be represented by $IV_{k,i}$ and CV_k, which is the same for all coefficients. Thus, it is sufficient that the header contains the common value of that packet and the information for individual values, i.e., changes with respect to the common value. This means that the header is of length $\log_2(q) + g \cdot t$ bits. Packet structures for RLNC and ReC are depicted in Fig. 2. The coefficients $c_{k,i}$s and CV_k are of length $\log_2(q)$ bits.

For b equal to 1, $IV_{k,i}$ is $b = 1$ bits long and $t = 1$ bits per coefficient should be transmitted. For $b = 2$, there are two approaches to determine $IV_{k,i}$:

Example 1: $IV_{k,i}$ is a value picked at random from $\mathbb{F}_{2^2}$ which means t is also equal to 2.

Example 2: The first bit is a random value from $\mathbb{F}_2$ and the second bit is the result of XOR operation of the first bit with a bit from the common value starting from the most significant bit for the first coefficient, shifted to the right by 1 for the next coefficient. The least significant bit will be skipped. Since the number of bits for a common value and for g are both a power of 2, this increases the number of possible values around the common value. Accordingly, in the second approach, the information for the second bit of the $IV_{k,i}$ is hidden in the first bit of that and in the common value, so there is no need to transmit the second bit. Thus, t is equal to 1.

2) Decoder: Decoding uses a similar mechanism as RLNC by applying Gaussian elimination, with one simple preprocessing step to regenerate the coefficients based on the ReC representation, i.e., taking the common value and applying the necessary bit flips in the appropriate location for each coefficient. Decoder finishes decoding by receiving g linearly independent packets.

3) Recoder: To generate a recoded packet, a coefficient is drawn at random from $\mathbb{F}_2$ is allocated to each packet in the node's buffer. The recoded packet is a combination of the last received packet and some of the buffered packets. Packets from the buffer are each selected at random with probability $1/2$ to be combined with the incoming packet, represented in Fig.3 with a $c_{k,j} = 1$ when they are selected to be combined. The recoded packet will be saved inside the node's buffer and will be transmitted to the next node. If the buffer is full, the new recoded packet replaces the oldest entry of the buffer.

The main objective of the proposed recoding scheme is to keep the same structure for headers like the one for the original coded packets. Moreover, doing the operations in $\mathbb{F}_2$ allows the recoder to have simple and fast procedures, whilst coding coefficients of new recoded packets are in $\mathbb{F}_q$. In this recoding approach, intermediate nodes generate recoded packets without decoding the arriving packets and without having g linearly independent coded packets.

There is an alternative recoding approach: recoding with decoding. The key difference comes when the recoder has received g independent packets. At this point, the recoder can decode the original packets and is able to generate new coded packets.

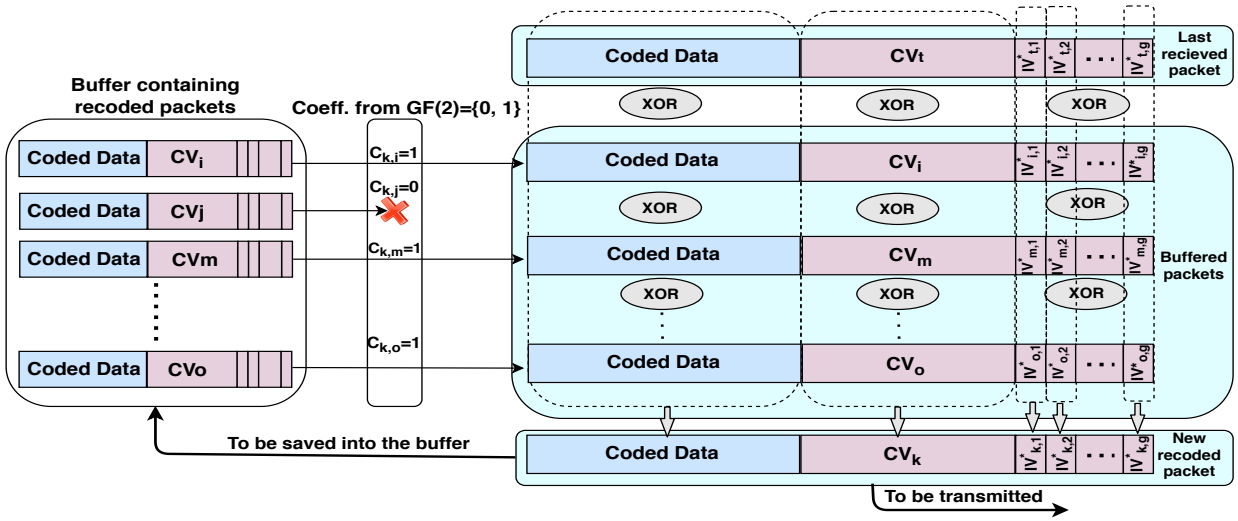

Fig. 3: Generating a recoded packet.

III. Overhead analysis

Let us define j as the number of extra packets sent to accomplish decoding, i.e., beyond the g coded packets that need to be transmitted as a minimum. The parameter p denotes the number of bytes in the payload of each packet and c is the number of bits per packet applied to send the information for coefficients to retrieve the data. Accordingly, the total overhead (O_T) is equal to:

$$O_T = (g + j) \times \lceil \frac{c}{8} \rceil + j \times p \quad [Bytes], \tag{2}$$

The total overhead comes from two sources: overhead due to the headers of coefficients i.e., c, and overhead due to the dependent packets, i.e., j. The parameter c for RLNC over $\mathbb{F}_q$ and ReC over $\mathbb{F}_q$ are equal to $g \cdot \log_2(q)$ and $\log_2(q) + g \cdot t$, respectively. Let us define header overhead reduction factor FO_H as a criteria to see how effectively ReC can decrease the overhead of headers compared to RLNC:

$$FO_H = \frac{g \cdot \log_2(q)}{g \cdot t + \log_2(q)}, \tag{3}$$

For high enough g values, $\log_2(q)$ would be negligible compared to $g \cdot t$. Thus, c for ReC approaches to $g \cdot t$. Hence, FO_H would be equal to $\log_2(q)/t$.

Example 3: Let us assume a generation size of $g = 128$ packets. In RLNC over $\mathbb{F}_{2^{16}}$, the header would be of length $128 \times 16 = 2048\, bits$. By applying $ReC(2^{16}, 2, 2)$, the header's length would be $16 + 128 \times 2 = 272\, bits$. This means *86.7 %* reduction in overhead due to the headers or equivalently, FO_H of 7.53. It will be shown in Section IV that the probability of having dependent packets is almost equal for these two cases.

Receivers finish decoding by having g linearly independent packets. In ReC, common values adopted from a large field size guarantee a low probability of dependent packets whilst overhead due to the headers is noticeably low thanks to the individual values adopted from a small field size. To evaluate the efficacy of the proposed idea, criteria **Dependency**

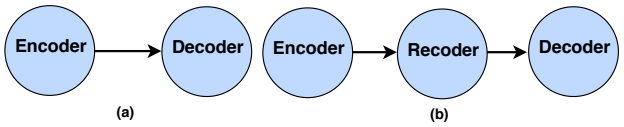

Fig. 4: System model for (a) one-hop network, and (b) two-hop network

Overhead Ratio (RO_D), **Header Overhead Ratio** (RO_H), and **Total Overhead Ratio** (RO_T), are used in this paper:

$$RO_D\% = \frac{j}{g} \times 100\%, \tag{4}$$

$$RO_H\% = \frac{\lceil \frac{c}{8} \rceil}{p} \times 100\% \tag{5}$$

$$RO_T\% = \frac{O_T}{g \times p} \times 100\% \tag{6}$$

IV. Numerical Results

A. Comparison Schemes

As a baseline comparison, simulation results for RLNC and Fulcrum codes are also plotted. In RLNC over $\mathbb{F}_q$ a coded packet is generated by linearly combining the original g packets by multiplying them by coefficients drawn at random from $\mathbb{F}_q$. Fulcrum codes [7] apply a precoding step at the source by using an expansion field over a large finite field. Fulcrum codes expand original g packets into $g + r$ coded packets, where r denotes the extra dimensions. Better probability of decoding is achieved with the increase of r at the expense of more overhead for signaling. After expansion, each packet is considered as a packet over $\mathbb{F}_2$. We used the current implementations in Kodo [9] for Fulcrum and RLNC in non-systematic form.

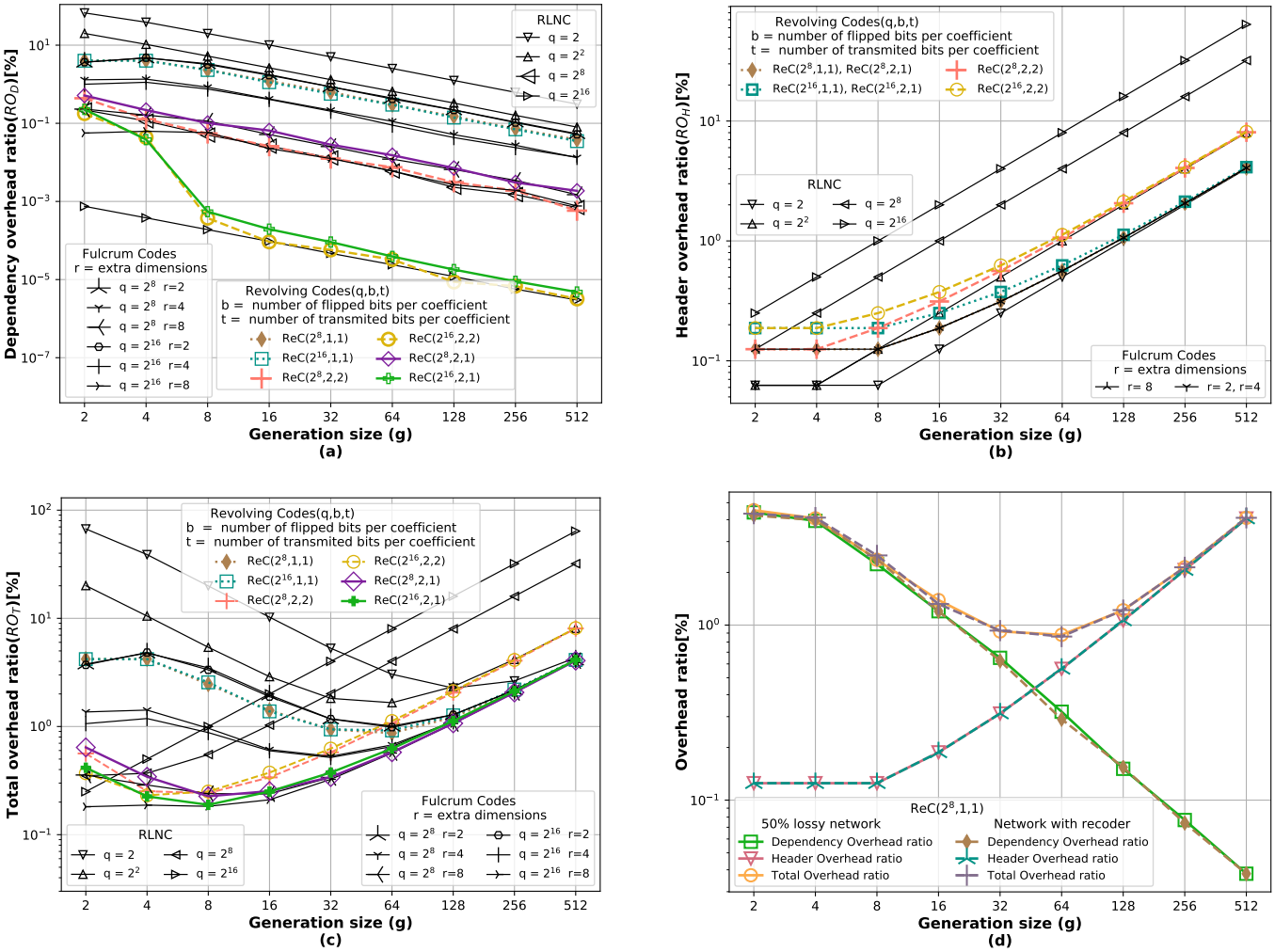

Fig. 5: Dependency, Header, and Total overheads and Overhead reduction versus generation size for packet size of 1.6 KB.

B. Simulations

We considered a packet size of 1.6 KB in our simulations, which is close to the MTU size of wireless networks [10]. For smaller payloads, the advantages of ReC are even more marked. The Kodo library [9] is used to implement ReC and each result is the average value over a minimum of 20000 runs. To obtain the results, two different system models are used as shown in Fig. 4 considering an erasure channel. Unless stated otherwise, the model in Fig. 4 (a) with one sender (encoder) and one receiver (decoder) is considered. For the analysis of recoding performance, we consider a two-hop network with one sender, one intermediate node, and one receiver with no direct link between sender and receiver.

Fig. 5 (a) demonstrates the RO_D using the system model shown in Fig. 4a. $ReC(2^8,2,1)$ and $ReC(2^{16},2,1)$ can reduce the RO_D by up to 99.4 % and 99.998 %, respectively, in comparison with RLNC over $\mathbb{F}_2$. Compared to Fulcrum codes, ReC can reduce RO_D by up to two order of magnitudes. This means that it has a high probability of decoding when receiving g independent packets. Since there are two conflicting forces, i.e., size of the field (q) and flipped bits (b), we observe almost equal dependency overheads are obtained for $ReC(2^{16},1,1)$ and $ReC(2^8,1,1)$ which means b is the dominating effect for those two field sizes.

Fig. 5 (b) shows the RO_H in the same network. In terms of RO_H, t is the dominating effect for high generation sizes in ReC. As shown in Fig. 5(b) RO_H for $ReC(q,b,t)$ converges toward the RO_H for RLNC over $\mathbb{F}_{2^t}$.

Fig. 5 (c) depicts the RO_T in such a network. Based on the discussed simulation results for RO_D and RO_H and as shown in Fig. 5 (c), $ReC(q,2,2)$ provides a reduction in total overhead, for different q values, in comparison with both RLNC $\mathbb{F}_q$ and RLNC $\mathbb{F}_{2^2}$ for a wide range of generation sizes. $ReC(2^8,2,2)$ can decrease RO_T by up to 74.81% and 97.36 % with respect to RLNC over $\mathbb{F}_{2^8}$ and over $\mathbb{F}_{2^2}$, respectively. Also, $ReC(2^{16},2,2)$ can reduce RO_T by up to 87.3 % and 98.1 % with respect to RLNC over $\mathbb{F}_{2^{16}}$ and over $\mathbb{F}_{2^2}$, respectively. $ReC(2^{16},1,1)$ reduces RO_T by up to 93.74 % in comparison with RLNC over $\mathbb{F}_{2^{16}}$. $ReC(2^8,2,1)$ and $ReC(2^{16},2,1)$ can reduce RO_T by up to 87.31 % and 93.56 % with respect to RLNC over $\mathbb{F}_{2^8}$ and over $\mathbb{F}_{2^{16}}$, respectively. In

terms of total overhead, ReC has comparable performance as Fulcrum with $r = 8$ extra dimensions.

Fig. 5 (d) demonstrates the overheads for $ReC(2^8, 1, 1)$ in a lossy network using the system model depicted in Fig. 4 (a) with the loss ratio of 50 % and in a network with a recoder as shown in Fig. 4 (b). We considered the overhead of the scheme at the decoder. Simulation results for $ReC(2^8, 1, 1)$ in a lossy network and in a network with a recoder in terms of RO_D are on average 1.7 % and 0.3 % different from RO_D for $ReC(2^8, 1, 1)$ in a simple network, respectively. Recoding and lossy network does not affect the RO_H. Since RO_T is determined by both RO_H and RO_D, the affect of lossy network and using a recoder on RO_T is negligible as demonstrated in Fig. 5(d). Accordingly, ReC shows a robust performance in a lossy network and in a network with a recoder.

Fig. 6 depicts the average number of transmitted packets from the encoder and the recoder to finish the decoding using the system model shown in Fig. 4 (b) for two different modes and for both of RLNC and ReC. In the first and the second modes, the link between the recoder and the decoder and the link between the encoder and the recoder are lossy, respectively, with the loss ratio of 50 %. The recoder's buffer size is half the generation size for g greater than 8, and equal to it for other cases. The results for recoder with decoder are demonstrated, as well. Recoder starts transmitting recoded packets by receiving the first packet. For the case of 50 % loss between the recoder and the decoder, the recoder generates two recoded packets by receiving each encoded packet. This extra redundancy helps with decreasing the unnecessary end-to-end redundancy. However, only the first one will be saved in the buffer. For generation size of 2, 50 % loss after the recoder and RLNC over $\mathbb{F}_2$, encoder and recoder need to transmit 6.2 and 12.4 packets on average, respectively, to finish the decoding process. These values are reduced to 3.0 and 6.0 packets by applying $ReC(2^8, 1, 1)$ which are close to the results achieved by RLNC $\mathbb{F}_{2^8}$, i.e., 2.5 and 5.1, respectively. Using a recoder that decodes results in optimal performance for ReC, the number of transmitted packets for ReC approaches the optimal performance by increasing the generation size.

V. Conclusion

A new network coding technique, called Revolving codes, is proposed. In addition to successfully reducing the total overhead of NC transmission, it provides a high probability of decoding. Moreover, it can be recoded using a simple, computationally efficient approach based on XOR operations and maintaining the code structure of ReCs. $ReC(q, 2, 1)$ can decrease the header overhead by a factor of up to $\log_2(q)$ for different q values in comparison with RLNC over $\mathbb{F}_q$. Compared to Fulcrum codes, Revolving codes decrease overhead caused by linearly dependent packets by as much as two orders of magnitude.

The benefit of ReC to decrease header and dependency overheads will be more significant in applications transmitting smaller payloads such as IoT and Industry 4.0. Let us assume a payload size of 100 B and a generation size of 64 packets.

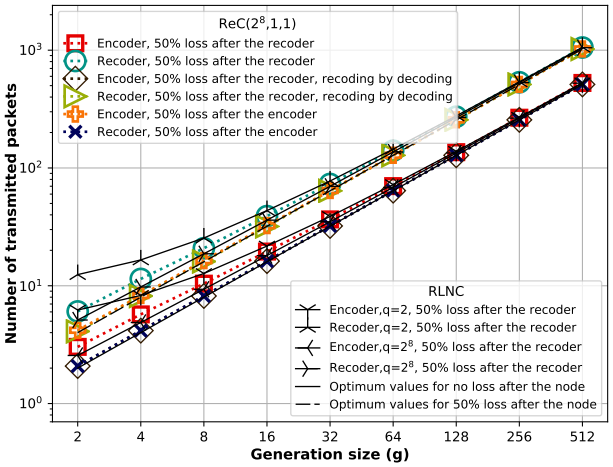

Fig. 6: The number of packets transmitted by the encoder and the recoder for $ReC(2^8, 1, 1)$ and packet size of 1.6 KB.

RLNC over $\mathbb{F}_{2^8}$ introduces a header overhead of 64 B, i.e., the header overhead ratio of 64 %. In contrast, $ReC(2^8, 2, 1)$ introduces a header overhead of 9 B, i.e., the header overhead ratio is 9 % with comparable performance in terms of dependency overhead. Future work will consider a mathematical analysis for ReC in order to estimate system performance under various finite field and network conditions.

Acknowledgments

This work was supported in part by the SCALE-IoT Project (Grant No. 7026-00042B) granted by the Danish Council for Independent Research, the Aarhus Universitets Forskningsfond Starting Grant Project AUFF- 2017-FLS-7-1.

References

[1] A. Paramanathan, M. V. Pedersen, D. E. Lucani, F. H. P. Fitzek, and M. Katz, "Lean and mean: network coding for commercial devices," *IEEE Wireless Communications*, vol. 20, no. 5, pp. 54–61, October 2013.

[2] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. on Info. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct 2006.

[3] D. Gligoroski, K. Kralevska, and H. Øverby, "Minimal header overhead for random linear network coding," in *IEEE International Conf. on Comms. Workshop (ICCW)*, June 2015, pp. 680–685.

[4] Y. Li, J. Zhu, and Z. Bao, "Sparse random linear network coding with precoded band codes," *IEEE Communications Letters*, vol. 21, no. 3, pp. 480–483, March 2017.

[5] J. Heide, M. V. Pedersen, F. H. P. Fitzek, and M. Medard, "A perpetual code for network coding," in *2014 IEEE 79th Vehicular Technology Conference (VTC Spring)*, May 2014, pp. 1–6.

[6] S. Wunderlich, F. Gabriel, S. Pandi, F. H. P. Fitzek, and M. Reisslein, "Caterpillar RLNC (CRLNC): A practical finite sliding window rlnc approach," *IEEE Access*, vol. 5, pp. 20 183–20 197, 2017.

[7] D. E. Lucani, M. V. Pedersen, D. Ruano, C. W. Sørensen, F. H. Fitzek, J. Heide, and O. Geil, "Fulcrum network codes: A code for fluid allocation of complexity," *arXiv preprint arXiv:1404.6620*, 2014.

[8] J. Heide and D. Lucani, "Composite extension finite fields for low overhead network coding: Telescopic codes," in *2015 IEEE International Conference on Communications (ICC)*, June 2015, pp. 4505–4510.

[9] M. V. Pedersen, J. Heide, and F. H. Fitzek, "Kodo: An open and research oriented network coding library," in *International Conference on Research in Networking*. Springer, 2011, pp. 145–152.

[10] Y. Li, S. Zhang, J. Wang, X. Ji, H. Wu, and Z. Bao, "A low-complexity coded transmission scheme over finite-buffer relay links," *IEEE Trans. on Comms.*, vol. 66, no. 7, pp. 2873–2887, July 2018.