IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 30, 2020, accepted April 20, 2020, date of publication April 22, 2020, date of current version May 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2989619

DSEP Fulcrum: Dynamic Sparsity and Expansion Packets for Fulcrum Network Coding

VU NGUYEN[1], (Graduate Student Member, IEEE), ELIF TASDEMIR[1], GIANG T. NGUYEN[1], (Member, IEEE), DANIEL E. LUCANI[2], (Senior Member, IEEE), FRANK H. P. FITZEK[1,3], (Senior Member, IEEE), AND MARTIN REISSLEIN[3,4], (Fellow, IEEE)

[1]Deutsche Telekom Chair, 5G Lab Germany, Technische Universität Dresden, 01062 Dresden, Germany
[2]Department of Engineering, Aarhus University, 8200 Aarhus, Denmark
[3]Centre for Tactile Internet With Human-in-the-Loop (CeTI), Technische Universität Dresden, 01062 Dresden, Germany
[4]School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ 85287-5706, USA

Corresponding author: Martin Reisslein (reisslein@asu.edu)

This work was supported in part by the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) as part of Germany's Excellence Strategy–EXC 2050/1–Project ID 390696704–Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop" (CeTI) of Technische Universität Dresden, in part by the SCALE IoT Project granted by the Danish Council for Independent Research under Grant DFF-7026-00042B, in part by the Aarhus Universitets Forskningsfond Starting Grant under Project AUFF-2017-FLS-7-1, in part by the Aarhus University's DIGIT Centre, the Ministry of Education and Training of Vietnam Project 911 under Grant 911/QD-TTg, and in part by the Ministry of National Education Turkey.

ABSTRACT Fulcrum coding combines a high-field outer Random Linear Network Coding (RLNC) that generates outer coding expansion packets with a small-field inner RLNC that combines the source packets and the outer coding expansion packets. This two-layer Fulcrum coding allows flexible decoding in receivers with heterogeneous computational capabilities. Fulcrum coding has so far only been studied for conventional dense RLNC, which randomly selects all coding coefficients, and only for a statically fixed number of outer expansion packets. However, the probability that the coding coefficient row of a newly received packet is linearly independent of prior received coding coefficient rows (a prerequisite for successful decoding) is highly dynamic. We propose to exploit the dynamics of this probability to reduce the computational complexity of Fulcrum coding. In particular, we vary the density of non-zero coding coefficients, i.e., equivalently, the sparsity of coding coefficients, and the number of outer expansion packets to keep the complexity low while maintaining a reasonably high decoding probability. We introduce the general principles of dynamic sparsity and expansion packets (DSEP) for Fulcrum coding as well as two specific example DSEP policies. Our evaluations indicate that DSEP Fulcrum can increase the encoding throughput tenfold and increase the decoding throughput 1.4 to 4.3 fold while achieving decoding probabilities that are typically less than 1% lower than the conventional Fulcrum decoding probabilities. We also find that DSEP achieves somewhat higher encoding and decoding throughputs than the CodornicesRq (Release 2.1) implementation of RaptorQ block coding for small blocks (generations) of source packets, while RaptorQ is substantially faster for large generation sizes. Furthermore, we develop and evaluate an elementary DSEP recoding mechanism that achieves a recoding throughput more than double the decoding throughput.

INDEX TERMS Computational complexity, heterogeneous devices, random linear network coding (RLNC), RaptorQ, recoding, sparsity, throughput.

I. INTRODUCTION

Random Linear Network Coding (RLNC) has the potential to greatly improve the performance of unreliable complex communication networks, including body area networks [3], cellular and radio access networks [4], [5], vehicular and wireless sensor networks [6], [7], and general wireless networks [8]–[12], as well as unreliable complex information technology infrastructures, such as data caching infrastructures [13]–[15]. One main hurdle that prevents the widespread adoption of RLNC in communication networks and information technology systems is the computationally

The associate editor coordinating the review of this manuscript and approving it for publication was Angelos Antonopoulos.

highly demanding matrix multiplication and matrix inversion required for RLNC decoding [16]–[19]. While the computational capabilities of network nodes are generally increasing and some senders and receivers have abundant computational capabilities [20], [21], a wide range of senders and receivers will continue to have limited computational capabilities for the foreseeable future. For instance, the emerging Internet of Things (IoT) paradigm features large numbers of low-cost senders and receivers with limited computational capabilities [22]–[25] that need to reliably communicate over error-prone wireless links and networks. On the other hand, due to emerging ultra-low-delay services and applications, there is a trend to push some service computing from the multi-access edge cloud (MEC) towards the clients and to let an ad hoc cloud of clients collaborate with the MEC [20], [21]. To make this involvement of clients in service computing feasible, it is critical to keep other workloads, e.g., from RLNC encoding and decoding, low.

The recently introduced Fulcrum RLNC [26] has addressed this high computation demand with a two-layer RLNC structure: An outer coding operates in a large Galois field $GF(2^h)$, e.g., with $h = 8$ (also referred to as high field), while an inner coding operates in the small $GF(2)$ (low field). While this two-layer Fulcrum RLNC is still computationally complex at the encoder, Fulcrum enables three possibilities for the decoding: A low-complexity inner decoding in $GF(2)$, a high-complexity outer decoding in the large $GF(2^h)$, and a combined decoding involving a mixture of operations in both fields. The Fulcrum coding introduced in [26] utilized a static number of expansion packets in the high field $GF(2^h)$ and employed dense coding in the sense that each coding coefficient was uniformly randomly drawn from the respective GF.

Recently sparse RLNC has emerged as a promising strategy for reducing the computational complexity of conventional (non-Fulcrum) RLNC encoding and decoding [27]–[32].

In this study, we advance the fields of Fulcrum coding and sparse RLNC by investigating sparse RLNC in the context of Fulcrum coding. We first examine static (fixed) levels of sparsity for the outer coding, the inner coding, as well as both the outer and inner coding in Fulcrum in Section III. We find that sparse inner coding combined with conventional dense outer coding, which we abbreviate to SIDO, achieves a good compromise between high coding throughput and high decoding probability.

Next, in Section IV, we introduce and evaluate dynamic SIDO that adapts the sparsity level of the inner coding according to the number of linearly independent coded packets at the receiver, i.e., the so-called receiver rank. Moreover, we dynamically vary the number of outer expansion packets that are utilized in the inner coding, resulting in the concept of dynamic sparsity and expansion packets (DSEP) Fulcrum, which is the main contribution of this article. We evaluate two example DSEP policies through extensive simulations. We find that DSEP Fulcrum vastly increases the encoding throughput (typically by over an order of magnitude) and achieves moderate decoding throughput increases (1.4 to 4.3 fold) compared to conventional Fulcrum coding. These throughput increases reflect the reductions in the encoding and decoding computational complexities achieved by DSEP Fulcrum. We confirmed that these complexity reductions are achieved at the expense of only slightly reduced decoding probabilities (typically less than 1% lower) compared to conventional Fulcrum. We also verified that these DSEP performance improvements are nearly maintained by practical feedback-free DSEP operation that assumes that the receiver rank is equal to the number of transmitted coded packets. Even for packet loss probabilities of 10% during network transport, the practical feedback-free DSEP operation achieves approximately 1.5 times higher decoding throughput and essentially the same decoding probabilities as conventional Fulcrum. We make the DSEP code base publicly available at https://github.com/nguyenvutud/DSEP.

TABLE 1. Summary of main notations.

n	Number of data packets in a generation
r	Number of outer coding expansion packets
$\mathcal{O}$	Set of original data pkts. and expansion pkts.
w	Sparsity level = Number of pkts. from $\mathcal{O}$ combined in inner coding, $w < (n+r)/2$ for sparse inner coding
μ	Number of expansion pkts. in $\mathcal{O}$ for DSEP, $0 \leq \mu \leq r$
δ	Nominal prescribed number of extra inner coded packets
i	Number of prior received lin. indep. coded packets, $0 \leq i \leq n-1$

II. BACKGROUND AND RELATED WORK

A. BACKGROUND ON FULCRUM CODES

The main notations are summarized in Table 1. Consider a sender transmitting a batch of n data packets, typically called a generation (or block), to the receivers. Fulcrum coding expands the set of original packets $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ into $n + r$ packets, as illustrated in Fig. 1. The r expansion packets are created with outer coding coefficients $c_{\ell,j}$, that are randomly selected by the encoder from $GF(2^h)$ [26], as

$$o_\ell = \sum_{j=1}^{n} c_{\ell,j} p_j, \quad \ell = 1, 2, \ldots, r. \tag{1}$$

These r expansion packets o_ℓ along with the n original data packets p_j are now treated as a new batch of packets $\{p_1, p_2, \ldots, p_n\} \cup \{o_1, o_2, \ldots, o_r\} = \mathcal{O} = \{o_j,\ j = 1, 2, \ldots, n, n+1, \ldots, n+r\}$, that will undergo the inner encoding in $GF(2)$. In particular, the inner encoding with the inner coding coefficients $\lambda_{\ell,j}$, that are randomly selected by the encoder from $GF(2)$, results in the inner coded packets

$$I_\ell = \sum_{j=1}^{n+r} \lambda_{\ell,j} o_j, \quad \ell = 1, 2, \ldots. \tag{2}$$

These inner coded packets I_ℓ are then sent throughout the network. The inner coded packets can be easily recoded at intermediate nodes and allow decoders to operate on either the inner or outer code, or a combination thereof, depending on the available computing power [36], [37]. A decoder that

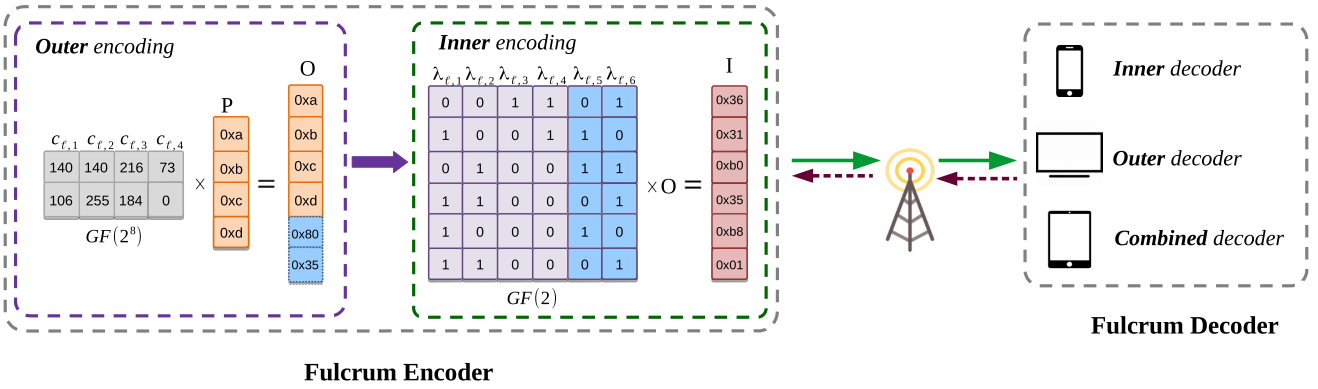


FIGURE 1. Illustration of Fulcrum outer and inner encoding [26] with $n = 4$ data packets and $r = 2$ expansion packets. The illustrated outer encoding is a systematic encoding that treats the original data packets as outer coded packets [15], [33]–[35] while the $r = 2$ expansion packets are linear combinations of the original data packets with outer coding coefficients $c_{\ell,j}$ from $GF(2^h)$. The $n + r = 6$ outer coded packets form the packet set $\mathcal{O}$, and these packets in the set $\mathcal{O}$ are linearly combined with inner coding coefficients $\lambda_{\ell,j}$ from $GF(2)$ to give the inner coded packets.

operates on the outer code has to perform particular steps to convert the inner code to the outer code before decoding in the high field $GF(2^h)$ [26], [37].

B. RELATED WORK

RLNC emerged from the compute-and-forward paradigm [38]–[41], which recodes packets in intermediate network nodes. RLNC can operate on blocks of source packets or a sliding window covering multiple source packets [42]–[47]. We develop DSEP Fulcrum as a block code that builds on RLNC block coding. Before reviewing related RLNC research, we briefly review high-performance block codes, which provide a useful comparison perspective for this study. High-performance block codes have typically low asymptotic computational complexity and achieve high packet decoding probabilities. A prominent line of recent high-performance block codes are Fountain codes [48]–[51] and their variations, such as the Luby Transform (LT) [52] and Raptor codes [53]–[55] (which are the basis for the Third Generation Partnership Project (3GPP) standard [56]). Block codes permit for recoding at intermediate network nodes in some specific networking scenarios, e.g., scenarios that permit so-called distributed LT coding [57], [58]. Generally, the coded packets could be decoded and then re-encoded in intermediate network nodes.

The RaptorQ code [59] is the basis for the forward error correction considered by the Internet Engineering Task Force (IETF) in the Request for Comments (RFC) 6330 [60] and is used by the Advanced Television Systems Committee (ATSC) 3.0 standard. RaptorQ consist of a pre-coding that generates intermediate symbols that are subsequently LT coded. The RaptorQ pre-coding consist of a low-density parity check code (LDPC) stage that operates in $GF(2)$ and a high-density parity check code (HDPC) stage that operates in $GF(2^8)$. RaptorQ has linear asymptotic computational complexity in the number n of source packets in a generation and achieves a packet decoding probability of 99% when decoding from n coded packets (i.e., without any received extra coded packets) [56], [59]–[61]. RaptorQ has been considered for a wide range of communications applications, e.g., [61]–[64].

RaptorQ allows for systematic coding, i.e., the source symbols can be sent in uncoded form, followed by the transmission of coded packets. More specifically, the source symbols are part of the encoding, and any combination of source and repair symbols can be used to recover missing source symbols. Systematic coding is important in practice: (*i*) to reduce end-to-end latency, since the source symbols can immediately be transmitted as they become available from the application, instead of waiting for all n source symbols to become available from the application and be encoded before transmitting any symbols; (2) because in most applications, there can be a mix of receivers, some supporting forward error correction (FEC) decoding, and some not, and thus it is crucial to be able to send the source symbols for the benefits of those receivers that do not have an FEC decoder. In contrast, Fulcrum RLNC and the DSEP Fulcrum RLNC developed in this article are non-systematic, sending all data in coded form. The development of systematic forms of Fulcrum RLNC and DSEP Fulcrum RLNC are important directions for future research.

Regarding recoding at intermediate network nodes, we note that a single RaptorQ decoding and re-encoding along a multi-hop path has the same complexity as a single RaptorQ encoding or decoding (without any packet header blowup). Moreover, since RaptorQ coding is a linear combination of intermediate symbols [60], one could specify a linear combination vector as part of the packet header. Such a linear combination vector could be represented with a packet header structure similar to the RLNC packet header structure [65], [66] and thus could keep track of linear recodings in intermediate network nodes. We employ the CodornicesRq (Release 2.1) implementation [67] of RaptorQ for our evaluations. CodornicesRq (Release 2.1) does

not include the outlined linear combination vector packet header structure. CodornicesRq (Release 2.1) can be used for encoding, decoding, or recoding of blocks of at least n symbols (when encoding, these are the source symbols; when recoding or decoding, these are the received symbols). The intermediate block is generated from these source symbols or received symbols. Once the intermediate block is generated, any symbol can be generated (when encoding, this is used to generate repair symbols to be sent; when recoding, this is used to generate recoded symbols for onward transmission; when decoding, this is used to generate missing source symbols).

One avenue for reducing the RLNC computational complexity is to consider small Galois fields [68]–[72], at the expense of reduced decoding probability due to higher probabilities of linearly dependent coded packets. The Fulcrum RLNC approach enables the flexible recoding and decoding in either a small Galois field (mainly at the expense of accumulating more coded packets to compensate for linear dependencies) or a large Galois field (incurring the high complexities), or a combination thereof [26]. An alternative approach to reduce computation complexity is systematic network coding [15], [33]–[35], which transmits the original packets in uncoded form and inserts a small number of coded packets into a generation to compensate for network transport losses. We employ systematic network coding in the outer coding of the sparse Fulcrum coding examined in this study.

The other main alternative to reducing the RLNC computational complexity is to utilize sparse coding coefficient rows, with only a relatively small prescribed number of non-zero coding coefficients. Sparse RLNC [73]–[78] has been examined in a range of contexts, including broadcast systems [79]–[81], data compression [82], and secure communications [83]. The decoding probability and delay for sparse RLNC have been analyzed in [84]–[88], while the tuning of the sparsity has been examined in [89]–[92], and sparsity for sliding window RLNC [42]–[47] has been studied in [93], [94]. To the best of our knowledge, sparse RLNC has not yet been examined in the context of the flexible Fulcrum RLNC approach. The present study builds on the prior work on sparse RLNC in non-Fulcrum contexts to devise and evaluate sparse forms of Fulcrum RLNC.

III. STATIC SPARSE FULCRUM CODING

A. ENCODING

The original Fulcrum encoding [26] (*i*) combines all original data packets with uniformly randomly generated outer coding coefficients $c_{\ell,j}$ in order to create the expansion packets, and (*ii*) combines all original data packets as well as all expansion packets with uniformly randomly generated inner coding coefficients $\lambda_{\ell,j}$ in order to create the inner coded packets. In contrast, we manipulate the outer and inner encoding by adjusting the number of non-zero coding coefficients, i.e., the sparsity of the outer and inner encoding.

Algorithm 1 Sparse Outer Fulcrum Encoding

Input: Gen. size n, data pkts. $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$, # of expansion pkts. r, sparsity level w
Output: Set of outer coded packets $\mathcal{O}$

```
O ← P
for ℓ = 1 to r do
    Set of considered data pkts. for encoding J ← ∅,
    initialize coded pkt. o_ℓ ← 0⃗, k ← 1
    while k ≤ w do
        j ← Random value (data pkt) from {1, 2, ..., n} \ J
        J ← J ∪ {j}
        c_{ℓ,j} ← Random value (cod. coeff.) from GF(2^h)
        o_ℓ ← o_ℓ ⊕ (c_{ℓ,j} p_j)
        k ← k + 1
    end while
    O ← O ⋃ {o_ℓ}
end for
```

Algorithm 2 Sparse Inner Fulcrum Encoding

Input: Gen. size n, # expansion pkts. r, set of orig. $\cup$ expansion. pkts. $\mathcal{O}$, sparsity level w,
Output: Inner coded packet I_ℓ

```
Set of considered pkts. J ← ∅, I_ℓ ← 0⃗, k ← 1
while k ≤ w do
    j ← Random value (pkt) from {1, 2, ..., n + r} \ J
    J ← J ∪ {j}
    I_ℓ ← I_ℓ ⊕ o_j
    k ← k + 1
end while
```

The sparse outer Fulcrum encoding, as summarized in Algorithm 1, combines only w, $w \leq n$, of the original data packets with outer coding coefficients $c_{\ell,j}$ that are randomly selected from $GF(2^h)$ to form a given expansion packet o_ℓ. (We neglect that the random selection gives with probability $1/2^h$ a zero and assume for simplicity that all w randomly selected coefficients are non-zero.) That is, effectively $n - w$ coding coefficients are zero, i.e., the corresponding $n-w$ data packets are not considered when forming the coded packet o_ℓ.

The sparse inner Fulcrum encoding, as summarized in Algorithm 2, combines only w, $w < (n + r)/2$, packets from the packet set $\mathcal{O}$ to form an inner coded packet I_ℓ. Conventional inner Fulcrum coding for an inner coded packet I_ℓ selects a random $GF(2)$ coding coefficient $\lambda_{\ell,j}$, $j = 1, 2, \ldots, n + r$, (which can be zero or one) for each of the $n + r$ packets in the set $\mathcal{O}$. On average, half of the $n+r$ coding coefficients $\lambda_{\ell,j}$ are zero, i.e., the corresponding packets are not considered in forming the inner coded packet. In Algorithm 2, we have modified the conventional algorithm for $GF(2)$ encoding [95] to only consider a uniform random subset of w, $w \leq (n+r)/2$, out of the $n+r$ packets of set $\mathcal{O}$ for forming a given inner coded packet I_ℓ. Effectively these w packets correspond to the packets with a coding coefficient $\lambda_{\ell,j}$ of one in conventional $GF(2)$ coding. Accordingly, the

sparsity definitions of the outer and inner Fulcrum coding are subtly different: For the outer encoding, w, $w \leq n$, is the number of considered coding coefficients, which are uniformly randomly drawn from $GF(2^h)$ (and could be zero with a minuscule probability); for the inner encoding, w, $w \leq (n+r)/2$, is the number of non-zero (i.e., one-valued) $GF(2)$ coding coefficients (whereby the uniform random selection of $n+r$ $GF(2)$ coefficients would on average result in $(n+r)/2$ one-valued coefficients).

In the following, we refer to the conventional encoding, which is accomplished for $w = n$ in the outer encoding and for $w = (n+r)/2$ in the inner encoding as "dense". Encodings with smaller w are referred to as "sparse". There are three variations of the sparse Fulcrum codes:

- *Sparse Inner–Sparse Outer (SISO)*: SISO applies sparse encoding for both the outer and inner encoding.
- *Sparse Inner–Dense Outer (SIDO)*: SIDO combines conventional dense outer RLNC encoding with sparse inner encoding with $w < (n+r)/2$ in Alg. 2.
- *Dense Inner–Sparse Outer (DISO)*: DISO combines sparse outer encoding with $w < n$ in Alg. 1 with conventional dense inner encoding, which has on average $(n+r)/2$ one-valued inner coding coefficients.

B. RECODING AND DECODING

The conventional Fulcrum recoding functionalities remain unchanged, i.e., typically, each intermediate node operates as an inner encoder. More specifically, when an intermediate node receives the $GF(2)$ inner coded packets, the node stores the packets in a buffer. The received packets are recoded with new randomly drawn $GF(2)$ coding coefficients to create new coded packets for transmission to the next hops. In particular, a randomly chosen half of the buffered packets are XORed with each other to create a recoded packet, as examined in Section V-C4. This recoding encompasses both the packet data and the coding coefficient vector in the packet header, so that the packet header size is not changed by the recoding and the existing techniques for the compact representation of the coding coefficient vectors, e.g., [66], can be applied.

The intermediate nodes can also use some other recoding techniques. For instance, an intermediate node can recreate the original code structure and generate the r additional dimensions that are missing in the inner code, if the node collects n linearly independent coded packets and then maps back to the high field $GF(2^h)$ in order to decode the data with outer decoding.

The decoder of sparse Fulcrum codes operates similarly to its Fulcrum counterpart. In order to decode the coded packets, the decoder can choose three types of decoding: inner, outer, or combined decoding [26]. The only modification is that a sparse Fulcrum decoder requires an additional parameter, namely the density w. When the decoder uses outer or combined decoding, it has exactly w non-zero outer coding coefficients to match the outer encoding of the encoder. Encoders and decoders can agree on the w value the same way they agree on other key parameters, such as the generation size n.

C. EVALUATION

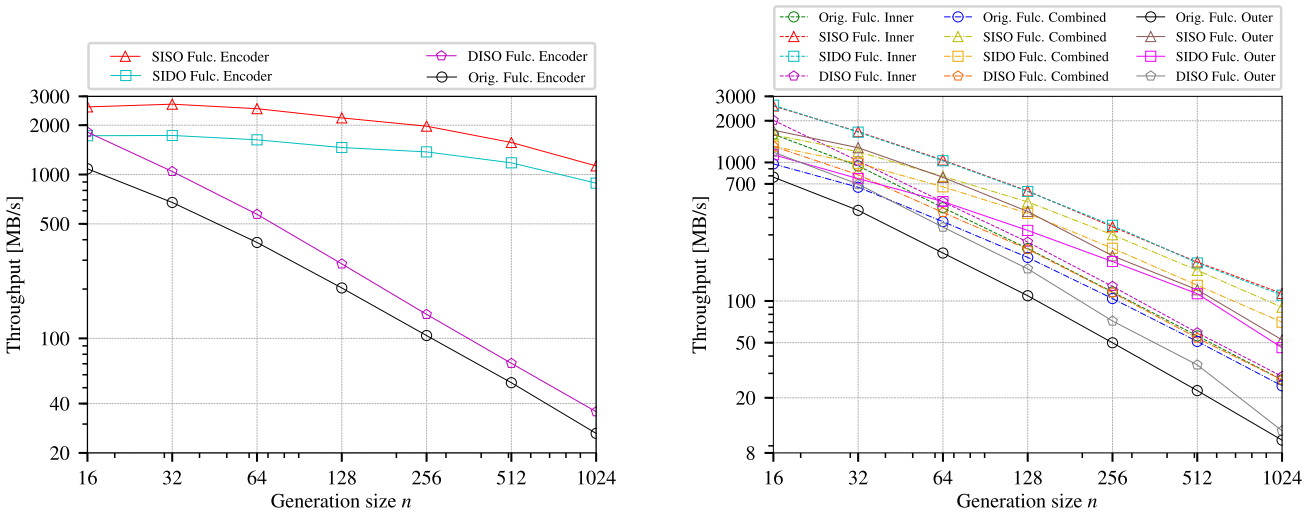
1) EVALUATION SETUP

We have implemented the three variations of sparse Fulcrum codes using the Kodo library (kodo-fulcrum version 7.0) [96], see code at https://github.com/nguyenvutud/DSEP. We have measured the encoding throughput and the decoding throughput as well as the decoding probability with the standard benchmarks available in the library. We have initially considered an end-to-end (one-hop) coding scenario, where the source node (sender) encodes and the destination node (receiver) decodes, without recoding at intermediate network nodes. The decoding employed a progressive decoder (on-the-fly version of Gauss Jordan algorithm [97]) that starts the decoding (coding coefficient elimination) with the first received packet [16], [19]. The encoding throughput is defined as the amount of payload data in a generation, i.e., $n\times$ data packet size, divided by the encoding computation time for $n+r$ coded packets, including the outer encoding and the inner encoding. The decoding throughput is defined as the amount of payload data in a generation, i.e., $n\times$ data packet size, divided by the decoding computation time for recovering the n original data packets. Generally, the energy consumption of RLNC encoding and decoding is linearly proportional to the computational complexity; the encoding and decoding throughput in turn is inversely proportional to the computational complexity [98], [99]. Thus, a high throughput indicates low energy consumption and vice versa. The detailed evaluation of the energy consumption of DSEP Fulcrum RLNC is left for future research.

The measurements were performed on a PC with Intel Core i5-4590 3.30 GHz CPU and 8 GB RAM. The data packet size was 1500 bytes which mimics the maximum size of Ethernet packets. The Galois field $GF(2)$ was used for the inner coding, while the outer coding was performed over the Galois field $GF(2^8)$. The number n of original packets in a generation was varied from $n = 16$ to 1024. The number of expansion packets was fixed to $r = 2$ and the sparsity level was fixed to $w = 5$. We conducted over 1000 independent replications for each evaluation scenario resulting in 95% confidence intervals that are too tight to be visible in the plots.

2) THROUGHPUT RESULTS

Fig. 2 shows the encoding and decoding throughputs. We observe from Fig. 2(a) that the two sparse Fulcrum variations with sparse inner coding, i.e., SISO and SIDO Fulcrum, achieve substantially higher encoding throughputs than original (dense) Fulcrum encoding and DISO Fulcrum. The encoding throughput differences between SISO and SIDO Fulcrum on one hand, and original and DISO Fulcrum on the other hand become particularly pronounced for large generation sizes n. The main reason for the strong impact of the sparse inner coding on the encoding throughput is the systematic outer coding, which conducts actual encoding operations only for the $r = 2$ expansion packets (the n original data packets are simply copied over to become systematic



(a) Encoding (b) Decoding

FIGURE 2. Encoding and decoding throughput [MByte/s] of static sparse Fulcrum variations and original Fulcrum for different generation sizes n for sparsity level $w = 5$ and $r = 2$ expansion packets.

"coded packets" [15], [33]–[35]). Thus, the overall computational effort for the encoding is dominated by the inner coding; even though the inner coding is conducted in the low-complexity $GF(2)$, all inner coded packets are combinations of the outer coded packets. Sparse inner coding combines a fixed number of $w = 5$ outer coded packets, while dense inner coding combines all n packets (more specifically, on average $(n+r)/2$ of the random $GF(2)$ inner coding coefficients will be one, i.e., $(n + r)/2$ outer coded packets will on average be combined to form a dense inner coded packet). Thus, the dense inner coding computation effort increases substantially for increasing generation size n. Generally, the RLNC encoding computational complexity scales on the order of $O(n^2)$ [18], leading to the nearly linear encoding throughput decrease with increasing n in the log-scale plot in Fig. 2(a). In contrast, the fixed sparsity level w gives nearly constant encoding throughput for increasing n in the log-scale plot in Fig. 2(a).

We note that for a non-systematic outer encoding, the relative contribution of the outer encoding to the overall computation effort for the encoding would increase and the encoding throughput would be relatively higher for the sparse outer coding schemes and relatively lower for the dense outer coding schemes.

Returning to the systematic outer coding, we observe from Fig. 2 that the impact of the sparse outer coding (comparison of SISO vs. SIDO as well as DISO vs. original Fulcrum) is significant, especially for small generation sizes n, e.g., DISO achieves nearly twice the encoding throughput of original Fulcrum for $n = 16$. For increasing generation size n, the impact of sparse outer coding shrinks, mainly because the outer encoding computation effort (for computing a fixed number of $r = 2$ expansion packets as linear combinations of the n data packets in $GF(2^8)$) shrinks relative to the inner encoding computation effort (for linearly combining on average $(n+r)/2$ outer coded packets in $GF(2)$) as the generation size n grows.

Turning to the decoding throughput in Fig. 2(b), we observe similar underlying trends as for the encoding throughput in Fig. 2(a). That is, SISO and SIDO Fulcrum achieve higher decoding throughputs than DISO and original Fulcrum, especially for large generation sizes n. For instance, for the combined and outer decoders, SISO and SIDO Fulcrum achieve more than double the decoding throughput of DISO and original Fulcrum for generation sizes of $n = 512$ and 1024. We also observe from Fig. 2(b) that inner decoding achieves higher decoding throughput than combined decoding, which in turn achieves higher decoding throughput than outer decoding. These differences are due to conducting the decoding in $GF(2)$, a mix of $GF(2)$ and $GF(2^8)$, and $GF(2^8)$, respectively.

Generally, we observe from Fig. 2 that sparsity can achieve much more pronounced throughput increases for encoding (more than an order of magnitude for large n) compared to decoding (only roughly a doubling). This is mainly because decoding needs to eliminate all n (or $n+r$) coding coefficients of an incoming coded packet, whereby the involved matrix inversion scales generally with $O(n^3)$. In contrast, the encoding only needs to linearly combine a fixed number of w packets, irrespective of the generation size n.

3) DECODING PROBABILITY RESULTS

Fig. 3 shows the decoding probability of a generation of $n = 128$ data packets as a function of the number of received extra coded packets, whereby zero received extra coded packets correspond to the receipt of n inner coded packets. We observe from Fig. 3 that combined decoding (which achieves the same decoding probabilities as outer

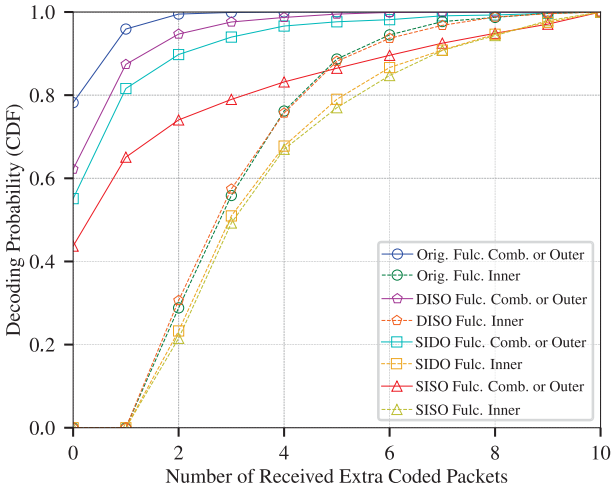


FIGURE 3. Decoding probability of generation of $n = 128$ data packets for different static sparse Fulcrum coding variations as a function of number of received extra coded packets (beyond n received packets) for $r = 2$ expansion packets and sparsity level $w = 5$.

decoding [26]) achieves significantly higher decoding probabilities than inner decoding, even when shifting the curves for inner decoding by $r = 2$ packets to the left to compensate for the fact that inner Fulcrum decoding require at least $n + r$ received inner coded packets (whereas combined and outer Fulcrum decoding requires at least n received inner coded packets [26]). The lower decoding probability of inner decoding is mainly due to the higher probability of linearly dependent coding coefficient rows in $GF(2)$ than in $GF(2^8)$.

We also observe from Fig. 3 that for a given decoder type, original Fulcrum achieves the highest decoding probabilities, followed by DISO, SIDO, and SISO. The dense $GF(2)$ inner coding coefficients have already a relatively high probability of linear dependent coding coefficient rows. Sparse inner coding reduces the number of non-zero $GF(2)$ coding coefficients, thus further increasing the probability of linearly dependent coding coefficient rows. SISO suffers from this high probability of linear dependent coding coefficient rows, which is only modestly compensated by the sparse outer coded expansion packets. The dense outer coding in SIDO compensates more strongly for the inner coding dependencies, resulting in a substantial increase of the combined decoding probability. DISO avoids the reduction of the linear independence by employing dense inner coding and only mildly suffers from the sparse outer coding, achieving slightly higher decoding probabilities than SIDO.

Overall, we conclude that SIDO represents a good compromise between high throughput (Fig. 2) and high decoding probability (Fig. 3) and we henceforth consider SIDO as the underlying sparse Fulcrum variation in the development and evaluation of DSEP Fulcrum. DSEP can be analogously applied to the other sparse Fulcrum variations.

IV. DYNAMIC SPARSITY COMBINED WITH DYNAMIC EXPANSION PACKETS

A. OVERVIEW

This section introduces dynamic sparse coding in Fulcrum in conjunction with dynamic tuning of the number of outer coding expansion packets. Based on the performance results for static sparse Fulcrum coding in Section III, we focus on SIDO, i.e., dynamic sparse inner coding in this section. As a foundation, we first analyze the probability of receiving an innovative coded packet that increments the number i of linearly independent coded packets at the receiver as a function of the sparsity level w of the encoding and the number r of expansion packets in Section IV-B. We then introduce the principle of dynamic sparse Fulcrum coding in Section IV-C and the principle of tunable expansion packets in Section IV-D. Subsequently, we analyze the sparsity level w, i.e., the number w of packets to combine in the inner encoding, as a function of a varying number of expansion packets μ in Section IV-E.

B. PROBABILITY OF LINEARLY INDEPENDENT CODED PACKET

Suppose that in conventional RLNC a receiver has received i, $i = 0, 1, 2, \ldots, n-1$, linearly independent coded packets (out of a generation of n packets, i.e., n linearly independent packets are required for decoding). The number i of received linearly independent coded packets is also referred to as the rank of the receiver coding coefficient matrix, or for brevity, the receiver rank. The probability that a newly received coded packet with density w/n (at most 0.5) is innovative (linearly independent with respect to the i received packets) has been bounded from below in [89] as

$$P_{innov.}^{RLNC}(i, n) \geq 1 - (1 - w/n)^{n-i}. \quad (3)$$

Due to the power law nature of the lower bound in Eqn. (3), the lower bound stays very close to one up to relatively large numbers i of received independent coded packets that approach the generation size n. For instance, for the conventional dense coding with $w/n = 0.5$ and a typical small generation size $n = 64$, the lower bound values are 0.999 for $i = n - 8$, 0.984 for $i = n - 6$, 0.937 for $i = n - 4$, 0.75 for $i = n - 2$, and 0.5 for $i = n - 1$. Larger generation sizes n give very similar dynamics; the lower bound of $P_{innov.}^{RLNC}$ depends mainly on the number $n - i$ of additional innovative packets that are missing to "fill the generation", i.e., to reach n received linearly independent coded packets.

For Fulcrum encoding with r expansion packets, the density is $w/(n+r)$ and the inner encoding matrix has dimension $(n + r) \times (n + r)$. The inner Fulcrum decoder needs $n + r$ linearly independent coded packets, i.e., the probability for a newly received coded packet to be innovative has to be evaluated for $i = 0, 1, 2, \ldots, n+r-1$ (prior) received linear independent coded packets:

$$P_{innov.}^{Fulc.,\ inn.}(i, n + r) \geq 1 - (1 - w/(n + r))^{n+r-i}. \quad (4)$$

On the other hand, the inner encoding matrix has dimension $(n) \times (n+r)$ for an outer or combined Fulcrum decoder. The outer or combined Fulcrum decoder needs n linearly independent coded packets. Thus, the probability for a newly received coded packet to be innovative has to be evaluated for $i = 0, 1, 2, \ldots, n-1$ (prior) received linear independent coded packets:

$$P_{innov.}^{Fulc.,\ out.}(i,n) \geq 1 - (1 - w/(n+r))^{n-i}. \tag{5}$$

We initially assume perfect feedback, i.e., that the sender has immediate accurate knowledge of the receiver rank i. In Section V-C, we will compare operating without feedback, i.e., assuming that the receiver rank i equals the number of transmitted coded packets, to operating with perfect feedback. According to Eqns. (4) and (5), the expected number of coded packets that is required to increase the number of linearly independent coded packets from i to $i+1$ can be upper bounded by $1/P_{innov.}^{Fulcr.}$.

We define δ as a nominal prescribed number of extra coded (overhead) packets. We suppose that δ is set sufficiently large to ensure the decoding of the entire generation. We will consider δ as an independent tuning parameter of the proposed DSEP policies, as examined in detail in Section IV-C. We proceed to briefly analyze a lower bound for δ by summing over an entire Fulcrum coded generation. Specifically, a receiver with an inner decoder requires the reception of $n+r$ linearly independent (inner) coded packets. The probability of the event of the reception of a linearly independent coded packet given i, $i = 0, 1, 2, \ldots, n+r-1$, (prior) received linear independent coded packets is given by $P_{innov.}^{Fulc.,\ inn.}(i, n+r)$, see Ineq. (4). The expected number of packet receptions required to receive a linearly independent packet is the inverse of $P_{innov.}^{Fulc.,\ inn.}(i, n+r)$. Thus, summing over the entire generation gives

$$n + r + \delta \geq \sum_{i=0}^{n+r-1} \frac{1}{P_{innov.}^{Fulc.,\ inn.}(i, n+r)}. \tag{6}$$

Following [89], we define the expected overhead for increasing the number of linearly independent packets at the receiver from i to $i+1$ as

$$\gamma(i, n+r) = \frac{1}{P_{innov.}^{Fulc.,\ inn.}(i, n+r)} - 1 \tag{7}$$

and note that the nominal prescribed number δ of extra coded packets should be larger than the actual number of extra coded packets needed for obtaining $n+r$ linearly independent coded packets at the receiver, i.e.,

$$\delta \geq \sum_{i=0}^{n+r-1} \gamma(i, n+r). \tag{8}$$

Analogously, a lower bound for δ can be obtained for outer or combined decoding which requires n linearly independent packets; thus, $\delta \geq \sum_{i=0}^{n-1} \gamma(i,n) = \sum_{i=0}^{n-1}(-1 + 1/P_{innov.}^{Fulc.,\ out.}(i,n))$. For the outer or combined decoding, the nominal prescribed number δ of extra coded packets would be defined with respect to n transmitted inner coded packets (whereas inner decoding considers δ with respect to $n+r$ transmitted inner coded packets). For consistency and ease of comparison, we follow the convention of the inner decoder for the rest of this paper, i.e., we consider the nominal prescribed number δ of extra (inner) coded packets with respect to $n+r$ (inner) coded packets.

C. PRINCIPLE OF DYNAMIC SPARSE FULCRUM CODING

Our goal is to define the density $w/(n+r)$ as a dynamic density that adapts according to the number i of received linearly independent coded packets. Thus, we define $w(i)$, $i = 0, 1, \ldots, n+r-1$, as the number of packets from the set $\mathcal{O}$ of original data packets and expansion packets that are to be combined for forming the next inner coded packet I_ℓ. We wish to express $w(i)$ as a function of the receiver rank i and the remaining Fulcrum encoding parameters, namely generation size n, number r of outer coding expansion packets, and parameter δ. There are two main avenues, namely setting the probability of a received innovative packet $P_{innov.}^{Fulc.}(i,n)$ to a constant or setting the overhead $\gamma(i, n+r)$ to a constant (that is held fixed as the receiver rank i varies). We pursue the analysis based on the overhead $\gamma(i, n+r)$ in Section IV-E. The overhead $\gamma(i, n+r)$ has been examined for conventional RLNC with different levels of sparsity w, GF field sizes, and generation sizes n in [95]. The results from [95] can be used to set a suitable δ for a particular encoding scenario.

D. PRINCIPLE OF DYNAMIC EXPANSION PACKETS

This section introduces the principle of dynamic expansion packets which adjusts the number of outer coding expansion packets that are included in the coding of inner coded packets. More specifically, a dynamic expansion packet protocol can exploit the number i of received linearly independent coded packets, i.e., the receiver rank i, to increase the number of included expansion packets so as to maintain a high probability of linear independent coded packets at the receiver.

The r expansion packets in the conventional Fulcrum encoding ensure a high probability of receiving linearly independent packets and increase the robustness against packet losses over error-prone networks. A small r is suitable when linearly dependent coded packets and packet losses are unlikely to occur. However, when there are frequent linearly dependent coded packets or packet losses, then a larger r will substantially increase the probability of linearly independent coded packets and the resilience against packet losses.

The main idea of dynamic expansion packets is to generate the inner coded packets without the expansion packets at the beginning of a generation and then to increase the number of expansion packets included in the formation of inner coded packets towards the end of a generation. When nearing the end of a generation, the decoder has collected a large number i of linearly independent coded packets, which decreases the probability of newly received coded packets being linearly independent.

More specifically, instead of including all n original data packets and all r expansion packets in the formation of inner coded packets, i.e., summing up to $n+r$ in Eqn. (2), we define a new variable μ, $0 \leq \mu \leq r$, for the number of expansion packets that are considered for the inner encoding. Formally, given the set of packets $\mathcal{O} = \{p_1, p_2, \ldots, p_n, o_{n+1}, o_{n+2}, \ldots, o_{n+r}\}$ produced by the outer encoding, the inner encoding includes a variable number μ, $0 \leq \mu \leq r$, of the expansion packets $o_{n+1}, o_{n+2}, \ldots, o_{n+r}$, i.e.,

$$I_\ell = \sum_{j=1}^{n+\mu} \lambda_{\ell,j} o_j, \tag{9}$$

For $\mu = 0$, no expansion packet is included in the linear combination forming the inner coded packet I_ℓ, i.e., for $\mu = 0$, the Fulcrum inner coding is equivalent to conventional RLNC in $GF(2)$.

E. SPARSITY LEVEL AS A FUNCTION OF NUMBER OF EXPANSION PACKETS μ AND RECEIVER RANK

When combining a dynamic sparsity level $w(i)$ with a dynamic number μ of expansion packets, the static (maximum) number r of expansion packets in Eqns. (4) and (5) needs to be replaced with the actual number μ of expansion packets that are considered for the encoding of the next inner coded packet (that seeks to increment the receiver rank from i to $i+1$). Thus, for Fulcrum inner decoding, where the number i of (prior) received linearly independent packets increases up to $n+\mu-1$ (whereby μ can ultimately increase up to r),

$$P_{innov.}^{Fulcr.,\ inn.}(i, n+r) \geq 1 - (1 - w/(n+\mu))^{n+\mu-i}. \tag{10}$$

For Fulcrum outer decoding,

$$P_{innov.}^{Fulcr.,\ out}(i, n) \geq 1 - (1 - w/(n+\mu))^{n-i}. \tag{11}$$

We derive the sparsity level, i.e., the number $w(i)$ of packets to combine in the inner decoding, so as to meet a fixed overhead of $\delta/(n+r)$ for each receiver rank i (as explained in Section IV-C), i.e., we posit

$$\gamma(i, n+r) = \frac{\delta}{n+r} \quad \forall i = 0, 1, \ldots, n+r-1. \tag{12}$$

We set the right-hand side of Eqn. (7) equal to the right-hand side of Eqn. (12) and utilize Inequality (10) for $P_{innov.}^{Fulc.,\ inn}(i, n+r)$ to obtain an upper bound for $w(i)$. Noting that the density should not exceed half of the packets in the set $\mathcal{O}$ considered for the inner encoding, i.e., should not exceed $(n+\mu)/2$, we set the sparsity level to

$$w(i) = (n+\mu) \min \left\{ \frac{1}{2},\ 1 - \sqrt[n+\mu-i]{1 - \frac{n+r}{n+r+\delta}} \right\}. \tag{13}$$

We round the $w(i)$ obtained from Eqn. (13) to the nearest integer and always set at least one coefficient, i.e., set $w(i)$ at least to one.

V. DSEP POLICIES

This section first introduces example policies for jointly dynamically adapting the number μ of expansion packets and the sparsity level w in Sections V-A and V-B. Then, Section V-C presents performance evaluation results for the introduced policies.

A. EXAMPLE POLICY: DYNAMIC SPARSITY WITH EXPANSION PACKETS REGION-BASED (DSEP-R)

For a given generation size n, maximum number r of expansion packets, and nominal overhead δ, the DSEP-R policy follows the region based approach from [31], referred to as DTEP approach in [2]. In particular, regions are specified by successive halving of the remaining set of missing packets. We define the cut-off receiver rank values

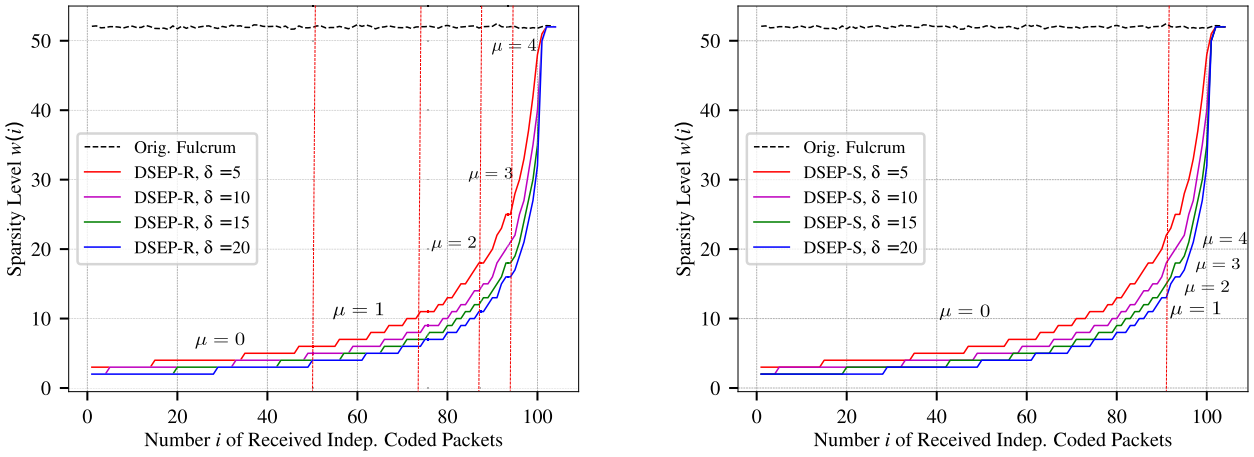
$$c(k) = \left\lfloor n \frac{2^k - 1}{2^k} \right\rfloor, \quad k = 1, 2, \ldots, r. \tag{14}$$

For instance, for $r = 4$, these cut-off values are $c(1) = \lfloor n/2 \rfloor$, $c(2) = \lfloor 3n/4 \rfloor$, $c(3) = \lfloor 7n/8 \rfloor$, and $c(4) = \lfloor 15n/16 \rfloor$ (which for $n = 64$ equal 32, 48, 56, and 60). Then, we set $\mu = 0$ for receiver ranks $i = 0, 1, \ldots, c(1)$ and adjust $w(i)$ according to Eqn. (13). Then, for receiver ranks $i = c(1)+1, \ldots, c(2)$, we set $\mu = 1$ and adjust $w(i)$ according to Eqn. (13), and so on. Finally, for receiver ranks $i = c(r)+1, \ldots, n+r$, we set $\mu = r$ and adjust $w(i)$ according to Eqn. (13). When an additional expansion packet is first added in, then the corresponding coding coefficient λ is set to one.

Fig. 4(a) illustrates the sparsity level $w(i)$ evaluated from Eqn. (13) for the DSEP-R policy for a range of nominal numbers δ of extra coded packets. We observe from Fig. 4(a) that a smaller δ increases the sparsity level w more aggressively than larger δ. Intuitively, a smaller δ strives to achieve the $n+r$ linear independent packets at an inner decoder or the n independent packets at a outer or combined decoder with a smaller number of transmitted inner coded packets. Accordingly, a smaller δ increases the number w of packets from set $\mathcal{O}$ that are combined in the inner coding more quickly as the number i of already received independent coded packets increases so as to ensure that the next received coded packet is with a higher probability linearly independent of all prior received coded packets.

B. EXAMPLE POLICY: DYNAMIC SPARSITY WITH EXPANSION PACKETS STEPPING UP (DSEP-S)

As noted in Section IV-B, the lower bound of $P_{innov.}^{RLNC}(i)$, see Eqn. (3), stays very close to one up to a receiver rank i (i.e., number i of prior received independent coded packets) that is quite close to the number n of packets in a generation. Typically, up to a receiver rank $i = n-8$, $P_{innov.}$ is very close to one; for a receiver rank of $i = n-6$ and higher, $P_{innov.}$ quickly drops significantly, reaching $P_{innov.} = 0.5$ for the receiver rank $i = n-1$. Thus, there is no need to include the Fulcrum expansion packets in the inner coding when the receiver rank i is typically eight or more below the generation size n. On the other hand, there is a potential to significantly



(a) DSEP-R

(b) DSEP-S with $\beta = 4$

FIGURE 4. Sparsity level $w(i)$, i.e., the number $w(i)$ of packets from the set $\mathcal{O}$ that are combined in the inner encoding, according to Eqn. (13) for DSEP Fulcrum as a function of the receiver rank i and the number μ of expansion packets for a generation of $n = 100$ data packets with a maximum number of $r = 4$ expansion packets for different nominal prescribed numbers δ of extra coded packets. The number μ of expansion packets follows the region-based DSEP-R policy in (a) and the stepping up DSEP-S policy with $\beta = 4$ in (b).

increase $P_{innov.}$ by including the Fulcrum expansion packets in the inner encoding when the receiver rank is within six or fewer coded packets of the generation size n.

Based on this insight, we specify the following stepping up adjustment strategy. We define a "beginning parameter" β, $0 \leq \beta \leq n - r$, that specifies the receiver rank from where on the expansion packets should be included in the inner coding. Specifically, if the receiver rank is below $n-r-\beta$, then the expansion packets are not included in the inner encoding. If the receiver rank is $i = n-r-\beta$, then the first expansion packet is included in the inner coding of the next packet (that seeks to increment the receiver rank to $i = n-r-\beta+1$). In order to ensure that this expansion packet is indeed included in the inner coding, the corresponding coding coefficient $\lambda_{\iota,n+1}$ is set to one, where ι denotes the number of the packet that is encoded at the sender after the receiver has achieved a rank of $n - r - \beta$. Each successive inner coded packet includes one additional expansion packet in the inner coding, until the number of expansion packets included in the inner coding reaches the available number r of expansion packets for receiver rank $i = n - r$. Each time that an additional expansion packet is for the first time included in the inner coding, the corresponding coding coefficient λ is set to one, as explained in more detail in the example of Eqn. (15) below. All of the subsequent packets are inner coded with r expansion packets. For a given value of the number of expansion packets μ, the number of non-zero coding coefficients $w(i)$ is evaluated according to Eqn. (13).

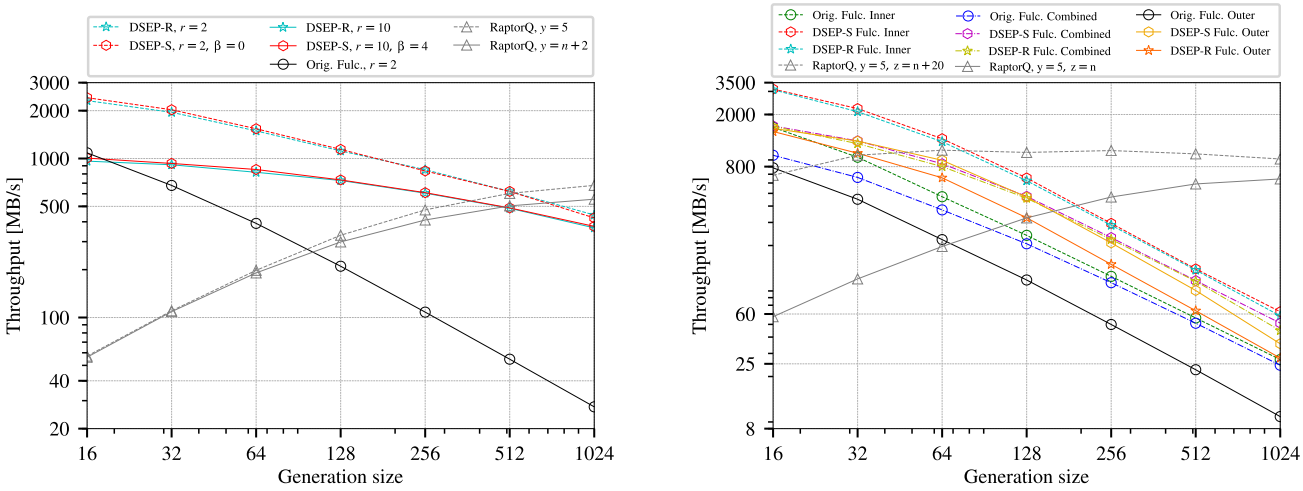
We illustrate the inner coding coefficients λ for an example with $n = 7$ data packets, $r = 3$ expansion packets, and $\delta = 3$ extra coded (overhead) packets in Eqn. (15) for a scenario where the number of transmitted coded packets equals the receiver rank. The original packets are $p_1, p_2, \ldots, p_7$, and the expansion packets are o_1, o_2, o_3 which are random linear combination of the original packets in $GF(2^h)$. Eqn. (15) presents the matrix of the inner coding coefficients λ with $n + \delta = 10$ rows ($\delta = 3$ extra coded packets) and $n + r = 10$ columns. For $\beta = 0$, the expansion is "turned on" for coded packet $n-r-\beta+1 = 5$ and the corresponding coding coefficient $\lambda_{5,8}$ is set to 1 (highlighted in red color in Eqn. (15)):

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ o_1 \\ o_2 \\ o_3 \end{bmatrix}. \quad (15)$$

The last three rows correspond to the $\delta = 3$ redundant (extra coded) packets, which are generated with the highest sparsity level $w = (n + r)/2$.

The number of non-zero coding coefficients $w(i)$, i.e., sparsity level, of this stepping up adjustment policy is illustrated for beginning parameter $\beta = 4$ and $r = 4$ expansion packets for different nominal prescribed numbers δ of extra coded packets in Fig. 4(b). We observe from Fig. 4(b) that similar to Fig. 4(a), smaller δ leads to higher $w(i)$. Moreover the comparison of Figs. 4(a) and (b) reveals that the different policies for increasing the number μ of considered expansion packets have only a very minor impact on the $w(i)$.

We note that the evaluation of the sparsity level w in both Figs. 4(a) and (b) is based on approximating the lower bound on the probability of linear independent packets $P_{innov.}^{Fulc.}$ in



(a) Encoder

(b) Decoder (inner, comb., and outer), $r = 2$ expan. pkts., $\beta = 0$

FIGURE 5. **Encoding and decoding throughput [MByte/s] of DSEP, original Fulcrum, and RaptorQ for different generation sizes *n*; fixed parameter $\delta = 20$.**

Inequality (4) as an equality. In order to examine the accuracy of this approximation, we have evaluated the Mean Square Error (MSE) between the probability values obtained from simulations and the probability values obtained from the lower bound along the trajectory of $w(i)$ as a function of the receiver rank i for the DSEP-R and DSEP-S policies, i.e., along the trajectories in Fig. 4. We found that the MSEs were in the range from 0.004 for $\delta = 5$ to 0.06 for $\delta = 20$. Thus, the approximation is reasonably close to serve in a practical DSEP protocol.

C. EVALUATION OF DSEP POLICIES

The section examines the throughput and decoding probability of the DSEP policies with the evaluation methodology from Section III-C1.

1) THROUGHPUT RESULTS

Fig. 5 compares the encoding and decoding throughputs of DSEP-S and DSEP-R with the original Fulcrum approach. For consistency of the comparison, we consider the encoding computation time for $n + r$ coded packets (and do not consider the encoding computation time for any extra inner coded packets, neither the nominal number of δ extra coded packets in the DSEP approaches, nor any extra coded packets required in the original Fulcrum approach for decoding a generation). We observe from Fig. 5(a) that both DSEP policies achieve substantially higher encoding throughputs than the original Fulcrum approach. For the small generation size $n = 16$, DSEP with $r = 2$ more than doubles the encoding throughput; while for the large generation sizes $n = 512$ and 1024, DSEP with both $r = 2$ and 10 increases the encoding throughput approximately tenfold compared to the original Fulcrum approach. These substantial encoding throughput increases are due to the sparse inner coding which is sped up by the reduced number of non-zero coding coefficients in the DSEP approaches, see DSEP, $\delta = 20$ curves in Fig. 4 compared to the original Fulcrum curves. Recall from the discussion in Section III-C2 that a fixed sparsity level w gives a nearly constant encoding throughput for increasing generation size n in the log-scale plot in Fig. 2(a). With dynamic sparsity, the sparsity level (encoding density) w is initially low when the receiver rank i is low, but the encoding density w approaches the density of conventional Fulcrum as the receiver rank approaches the generation size n. Thus, the dynamic sparsity encoding throughput curves in Fig. 5(a) fall essentially between the sparse inner coding (SISO and SIDO) curves and the original Fulcrum curve in Fig. 2(a).

We observe from Fig. 5(b) that the DSEP policies with $r = 2$ significantly increase the decoding throughput compared to the original Fulcrum approach. For each type of decoder (inner, combined, and outer), DSEP-R increases the decoding throughput roughly between 1.7 to 3 times for generation sizes of $n = 128$ and larger compared to the original Fulcrum, while DSEP-S increases the decoding throughput between 1.7 to 4.3 times. These decoding throughput increases with DSEP are again due to the reduced number of non-zero coding coefficients (see Fig. 4). However, the decoding throughput increases with DSEP compared to the original Fulcrum in Fig. 5(b) are not as large as the corresponding encoding throughput increases in Fig. 5(a) since the decoder needs to eliminate all n (or $n + r$) coding coefficients.

For an additional comparison perspective of the DSEP Fulcrum encoding and decoding throughputs, we compare with the RaptorQ throughputs. Generally, encoding and decoding throughputs depend on the code implementation. We consider the CodornicesRq (Release 2.1) implementation of RaptorQ [67], [100], which was created under the leadership of M. Luby, who developed LT coding [52] and was

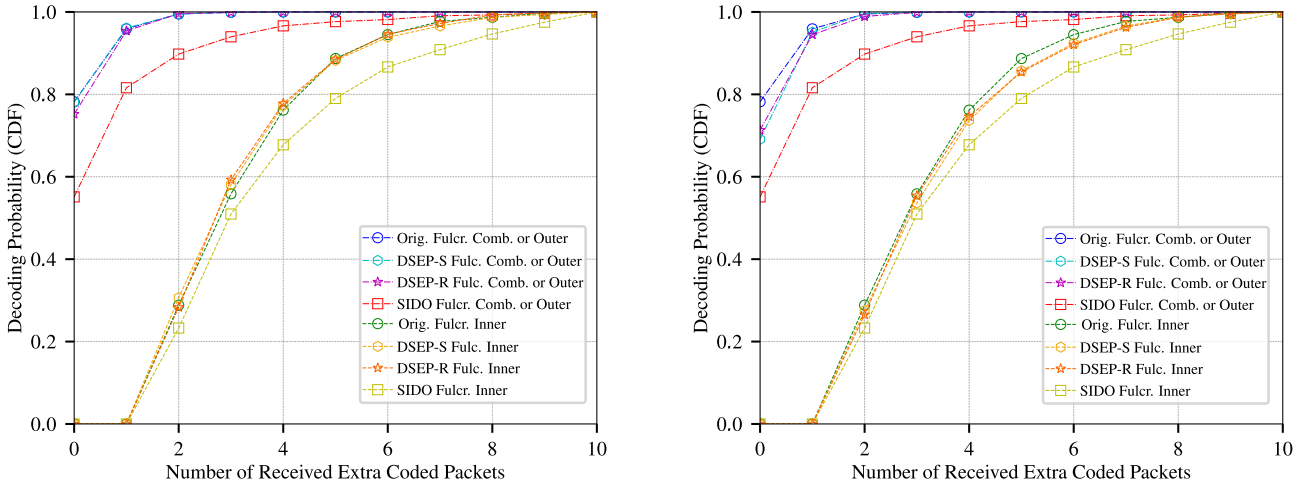
TABLE 2. Decoding throughput [MByte/s] of DSEP policies, original Fulcrum, and RaptorQ for different generation sizes n and parameters δ for inner, combined, and outer decoder.

Policy	Decoder Type	$n = 32$		$n = 128$		$n = 512$	
		$\delta = 5$	$\delta = 20$	$\delta = 5$	$\delta = 20$	$\delta = 5$	$\delta = 20$
DSEP-R, $r = 2$	Inner	1529	2101	435	624	108	129
	Comb.	961	1198	349	459	90	106
	Outer	785	1012	217	325	51	63
DSEP-R, $r = 10$	Comb.	692	890	276	364	80	94
DSEP-S, $r = 2, \beta = 0$	Inner	1622	2218	454	655	110	132
	Comb.	1005	1258	360	474	93	107
	Outer	1065	1252	325	468	74	90
DSEP-S, $r = 10, \beta = 4$	Comb.	556	683	265	323	82	91
Original Fulcrum, $r = 2$	Inner	943		239		54	
	Comb.	662		206		49	
	Outer	451		109		21	
Original Fulcrum, $r = 10$	Comb.	405		160		46	
RaptorQ, $y = 5, z = n + 1$		132		369		638	
RaptorQ, $y = n + 2, z = n + 1$		126		327		517	
RaptorQ, $y = 5, z = n + 20$		1017		1110		1029	
RaptorQ, $y = n + 2, z = n + 20$		840		875		798	

instrumental in developing Raptor and RaptorQ [50], [60]. CodornicesRq prefers symbol sizes that are multiples of 64 byes; therefore, we set the packet size to 1536 bytes for the RaptorQ evaluations.

Fig. 5(a) shows the RaptorQ encoding throughput when generating $y = 5$ and $y = n + 2$ repair symbols. We observe from Fig. 5(a) that a smaller number y of repair symbols leads to a slightly higher encoding throughput. Comparing DSEP and RaptorQ, we observe from Fig. 5(a) that DSEP achieves substantially higher encoding throughput than RaptorQ for small generation sizes n; while for generation sizes of $n = 512$ and larger, RaptorQ achieves higher encoding throughput than DSEP. In interpreting these encoding throughput results it is important to keep in mind that RaptorQ performs systematic encoding, i.e., the n source packets can first be sent in uncoded form as they become available from the application, followed by the y coded repair symbols. In contrast, DSEP Fulcrum performs non-systematic coding, i.e., the dense outer coding must be fully completed before the first inner coded packet can be generated (according to the sparse inner coding Algorithm 2) and transmitted; subsequent inner coded packets can be successively generated (with Alg. 2) and transmitted. For generation sizes above 1024, i.e., outside our plotted range, RaptorQ has vastly superior encoding throughput, as examined in detail in [100], due to a linearly increasing asymptotic encoding computational complexity in n; in contrast, the computational complexity of RLNC encoding scales asymptotically with $O(n^2)$. The implementation optimization of CodornicesRq has focused on large values of n. Optimized implementations of RaptorQ for small values of n are possible and are an interesting direction for future research. The decoding throughput for $z = n + 20$ examined next in Fig. 5(b) gives an indication that CodornicesRq (Rel. 2.1) has generally substantial optimization potential. Fig. 5(a) indicates that for a moderately small generation size of $n = 128$, DSEP Fulcrum achieves an encoding throughput above 600 MByte/s, i.e., roughly twice times the encoding throughput of CodornicesRq. DSEP Fulcrum inherits the very small static computational overhead component in Kodo RLNC [101].

Fig. 5(b) and Table 2 give the RaptorQ decoding throughput with $y = 5$ or $y = n + 2$ repair symbols and utilizing $z = n$ or $n + 20$ received coded packets for decoding. The DSEP and original Fulcrum decoding may require a few extra coded packets (beyond n coded packets) for successful decoding, as examined in detail in Section V-C2, however, the DSEP and Fulcrum decoding throughput does not increase with the availability of additional coded packets. We observe from Fig. 5(b) that for the very small generation size $n = 16$, DSEP achieves substantially higher decoding throughput than RaptorQ. Specifically, for $n = 16$, the DSEP decoding throughput is roughly double the RaptorQ decoding throughput if RaptorQ has 36 coded packets available for decoding. If RaptorQ has only $n = 16$ coded packets available for decoding, then the decoding throughput drops roughly by a factor of ten to around 60 Mbyte/s. The decoding throughput of RaptorQ thus greatly increases with the number of extra coded packets (beyond the number n of packets in a generation). These results indicate that the CodornicesRq computation for encoding and decoding with low overhead for all generation sizes n will be dramatically faster once the implementation has been optimized for the case when the number of symbols used to generate the intermediate block is close to n. In particular, the dramatic decoding throughput increase for the small generation size $n = 16$ with 20 extra coded packets indicates that there is significant potential for speeding up the RaptorQ computations for encoding and decoding with low overhead. On the other hand, for the large $n = 1024$ generation size, RaptorQ achieves more than ten times the DSEP decoding throughput. For large generation sizes above $n = 1024$, the RaptorQ decoding throughout is also much higher than the DSEP Fulcrum decoding throughput as RaptorQ has a linear asymptotic decoding complexity [100], while the underlying asymptotic computational complexity is $O(n^3)$ for RLNC decoding. Overall, we can thus conclude that for small generation sizes n, which are well-suited for small data sets and low-latency low-bandwidth communication, DSEP Fulcrum achieves encoding and decoding throughputs that are higher than for the current CodornicesRq implementation of RaptorQ. On the



(a) $\delta = 5$ (b) $\delta = 20$

FIGURE 6. Decoding probability of DSEP policies compared to original Fulcrum and SIDO Fulcrum for the combined or outer decoder and the inner decoder as a function of the number of received extra inner coded packets beyond the number of $n = 128$ of original source packets in a generation. Fixed parameters: $r = 2$ outer coding expansion packets, sparsity level $w = 5$ for SIDO, $\beta = 0$ for DSEP-S, and network without packet losses $\epsilon = 0$.

other hand, for large generation sizes n, DSEP Fulcrum gives lower throughputs than RaptorQ due to the higher asymptotic complexity of the underlying RLNC encoding and decoding in DSEP Fulcrum.

We return to the throughput evaluation of DSEP Fulcrum to examine the impacts of the different DSEP policies and the parameter δ. We first observe from Figs. 5(a) and (b) that the two DSEP policies give generally very similar encoding and decoding throughputs. Only for the outer decoder do we observe that DSEP-S gives somewhat higher decoding throughput than DSEP-R. In order to further investigate these differences between the DSEP policies and to examine the impact of the parameter δ, we compare in Table 2 the decoding throughputs of the two DSEP policies for a range of generation sizes n and two parameter δ values. We observe from Table 2 that DSEP-S with $r = 2, \beta = 0$ achieves very slightly higher decoding throughputs than DSEP-R with $r = 2$ for inner and combined decoding, while for outer decoding the DSEP-S decoding throughput is up to approximately 1.5 times higher than for DSEP-R. We also observe from Table 2 that this result is reversed for $r = 10$; in particular, DSEP-R with $r = 10$ achieves higher decoding throughputs than DSEP-S with $r = 10, \beta = 4$ (for brevity, only the combined decoding throughputs are given in Table 2, but this result holds for inner and outer decoding as well).

For $r = 2, \beta = 0$, most DSEP-S coded packets include fewer outer expansion packets than the DSEP-R coded packets. In particular, DSEP-S keeps the number of included outer expansion packets at $\mu = 0$ up until the coded packets are prepared for increasing the receiver rank above $n - r - \beta$. In contrast, DSEP-R includes outer expansion packets earlier, according to the region based policy, see illustration in Fig. 4. The processing of these outer expansion packets in the decoder requires computationally demanding $GF(2^8)$ operations. More specifically, with μ outer encoding expansion packets, a given inner coded packet has on average contributions from $\mu/2$ outer expansion packets, i.e., packets with high field $GF(2^8)$ coding coefficients. Thus, when mapping back in the outer decoder [26], there are on average $\mu/2$ coding coefficient rows with $GF(2^8)$ elements. These high field coding coefficient rows greatly increase the multiplication operations in the decoding process, lowering the decoding throughput. For $r = 10, \beta = 4$, this dynamic is reversed so that DSEP-R coded packets include fewer outer expansion packets than the DSEP-S coded packets.

We also observe from Table 2 that a smaller δ parameter reduces the decoding throughput, whereby the impact of a smaller δ becomes slightly less pronounced for the large $n = 512$ generation size. As elaborated in the decoding probability evaluation in Section V-C2, it is generally not necessary to encode, transmit, and decode the nominal prescribed number of δ extra coded packets. However, a smaller δ parameter increases the encoding density w (see Eqn. (13)), which in turn increases the encoding and decoding computational effort. Generally, for small generation sizes n the computation effort is mainly a static overhead; however, for large n the $O(n^3)$ proportionality of the decoding computations becomes dominant [18]. Nevertheless, we observe from Table 2 that for the small $\delta = 5$, both DSEP policies with $r = 2$ achieve 1.7 fold or higher decoding throughput increases for generation sizes of $n = 128$ and 512 compared to original Fulcrum, while the decoding throughput increases with $r = 10$ are about 1.3 for $n = 128$ and over 3.3 for $n = 512$.

2) DECODING PROBABILITY RESULTS

Fig. 6 shows the decoding probabilities (when having received up to and including the ρth extra coded packet, i.e., a total of $n + \rho$ coded packets) corresponding to the scenarios

TABLE 3. Decoding probability (in percent) for combined or outer decoder as a function of the number r of outer expansion packets and the number of received extra inner coded packets; fixed parameters: generation size $n = 128$, and $\beta = 4$.

Policy	$r = 2$				$r = 4$		$r = 6$		$r = 10$	
	0	1	2	3	0	1	0	1	0	1
Orig. Fulc.	77.16	95.88	99.39	99.91	93.43	99.74	98.05	99.967	99.56	99.998
DSEP-R, $\delta = 5$	76.64	96.25	99.43	99.98	93.23	99.68	98.03	99.937	99.40	99.993
DSEP-R, $\delta = 20$	71.82	94.54	99.10	99.84	85.68	98.89	88.98	99.470	94.32	99.877
DSEP-S, $\delta = 5$	76.60	96.02	99.43	99.94	93.48	99.72	98.14	99.957	99.44	99.996
DSEP-S, $\delta = 20$	70.55	94.24	99.10	99.86	86.64	99.07	92.43	99.756	95.43	99.910

investigated in the preceding throughput section. We observe from Fig. 6 that for $\delta = 5$, the DSEP policies achieve approximately the same decoding probabilities as the original Fulcrum approach; while for $\delta = 20$, the DSEP decoding probabilities are very slightly lower than the original Fulcrum decoding probabilities. We also observe that SIDO Fulcrum gives generally lower decoding probabilities than DSEP and original Fulcrum. These decoding probability results confirm that for a small δ parameter setting, the dynamic adaptation of the sparsity level $w(i)$ of the encoding as the receiver rank i increases is effective in maintaining nearly the same level of linear independence of the coding coefficients and thus approximately the same high decoding probabilities across the entire range of prior received coded packets (receiver ranks) i, $i = 0, 1, \ldots$ (up $n-1$ for combined or outer decoder; up to $n + r - 1$ for inner decoder). Thus, the DSEP policies ensure nearly the same effective number of received linearly independent coded packets and approximately the same high decoding probabilities for a given number of received extra coded packets as the original Fulcrum approach. In contrast, the static (fixed) sparsity level of the SIDO approach suffers from lowered decoding probabilities as the receiver rank i approaches the generation size n (see Eqn. (3)) and therefore requires more extra coded packets than DSEP and original Fulcrum to decode a generation.

Table 3 further examines the decoding probability for combined or outer decoding for $r = 2$, 4, 6, and 10 outer expansion packets and for 0, 1, 2, and 3 received extra coded packets. For $r = 4$, 6, and 10, the decoding probabilities are one when receiving two or more extra coded packets and the columns for 2 and 3 extra coded packets are therefore omitted from the table. We observe from Table 3 that for the small $\delta = 5$, the DSEP decoding probabilities are equivalent to or at most 0.6% lower than the original Fulcrum decoding probabilities. Overall, for scenarios with high decoding probabilities of 95% or higher for the original Fulcrum, both DSEP policies with $\delta = 5$ are within 0.2% of the original Fulcrum decoding probabilities.

To put these Fulcrum DSEP RLNC decoding probability results further in perspective, we compare with the decoding probabilities of RaptorQ, which are 99% without any received extra coded packets and 99.99% for one received extra coded packet [56], [59]–[61]. We observe from Table 3 that for $r = 10$ outer coding expansion packets, both DSEP approaches with $\delta = 5$ achieve the 99% decoding probability without any received extra coded packets. Similarly, for $r = 10$ and one received extra coded packets, both DSEP-R and DSEP-S achieve the 99.99% decoding probability with $\delta = 5$. Thus, we conclude that DSEP with small δ parameter and moderately large number r of extra coded packets achieves similarly high decoding probabilities as RaptorQ (with the same number of received coded packets). We note that the higher number of $r = 10$ outer coding expansion packets reduces the throughputs somewhat, as shown in Fig. 5(a) and Table 2; however, for small generation sizes n, DSEP with $r = 10$ still achieves higher encoding throughput than RaptorQ. The decoding throughput comparison depends on the number of extra coded packets available for RaptorQ decoding: with one extra packet ($z = n + 1$), DSEP decodes faster; while for twenty extra packets ($z = n + 20$), CodornicesRq decodes faster.

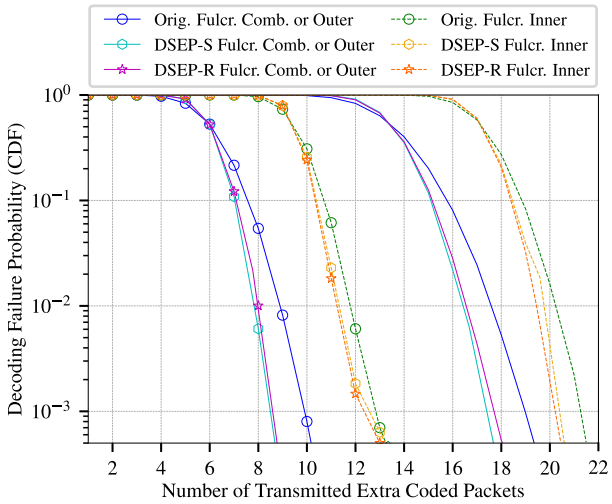


FIGURE 7. Packet loss evaluation: Decoding failure probability of DSEP policies compared to original Fulcrum as a function of the number of transmitted extra inner coded packets beyond the $n = 128$ original source packets for packet erasure probabilities $\epsilon = 0.05$ (with point markers and lines) and 0.1 (without point markers, only lines). Fixed parameters: $r = 2$ outer coded expansion packets, $\beta = 4$ for DSEP-S, $\delta = 5$, combined or outer decoder as well as inner decoder.

3) IMPACT OF FEEDBACK AND PACKET LOSSES

This section evaluates the impact of feedback about the actual receiver rank i to the encoder and the impact of losses (erasures) of transmitted coded packets during the network transport. Initially, we focus on packet losses and continue to consider DSEP without feedback and compare in Fig. 7

TABLE 4. Decoding throughput [MByte/s] of DSEP policies with and without perfect feedback and original Fulcrum for inner, combined, and outer decoder, as well as RaptorQ for different packet erasure probabilities ϵ; fixed parameters: DSEP parameters $\delta = 5$, $\beta = 4$, as well as RaptorQ parameter $y = n\epsilon/(1-\epsilon)$.

Policy	Decoder	$n = 32$				$n = 128$			
		$\epsilon = 0.05$		$\epsilon = 0.1$		$\epsilon = 0.05$		$\epsilon = 0.1$	
		$r = 2$	$r = 10$	$r = 2$	$r = 10$	$r = 2$	$r = 10$	$r = 2$	$r = 10$
DSEP-R No feedback	Inner	1465	1090	1362	1030	393	390	371	364
	Comb.	890	621	850	570	308	247	303	237
	Outer	712	618	678	522	206	176	194	156
DSEP-R Perf. feedback	Inner	1518	1126	1503	1123	438	385	436	384
	Comb.	926	698	925	678	346	276	340	273
	Outer	783	648	778	643	240	187	238	185
DSEP-S No feedback	Inner	1514	1151	1437	1103	409	380	389	362
	Comb.	872	523	830	509	331	246	312	236
	Outer	741	493	688	491	262	192	231	126
DSEP-S Perf. feedback	Inner	1581	1152	1561	1137	430	376	428	375
	Comb.	933	563	930	557	351	256	348	254
	Outer	846	548	782	498	305	213	304	210
Orig. Fulc.	Inner	856	697	852	689	235	208	234	205
	Comb.	584	381	575	377	205	151	205	157
	Outer	340	338	388	320	107	84	107	76
RaptorQ, $z = n + 1$		132		131		366		370	
RaptorQ, $z = n + 20$		1043		1039		1097		1088	

the cumulative decoding failure probability for the two DSEP policies against the original Fulcrum approach. We plot the log-scaled cumulative decoding failure probability of a generation of n original source packets as a function of the number τ of transmitted extra coded packets beyond n coded packets, i.e., the probability that the decoding fails when having transmitted up to and including the τth extra coded packet. We observe that in the regions with small decoding failure probabilities below around 0.6, the DSEP policies achieve slightly smaller decoding failure probabilities than the original Fulcrum approach. The underlying reason for this small difference in the decoding probabilities is mainly due to a subtle difference in the generation of the non-zero coding coefficients in the inner encoding. In the original Fulcrum approach, the conventional dense inner encoder generates an average number of $(n + r)/2$ non-zero coding coefficients. Thus, for a given coded packet, the actual number of non-zero coding coefficients may randomly vary around the mean of $(n+r)/2$. In contrast, when the number of transmitted packets in the DSEP policies without feedback exceeds n, then the number of non-zero coding coefficients is deterministically set to $(n + r)/2$, see Eqn. (13). The combination of a few more packets from the outer encoding in the inner encoding can very slightly increase the decoding success probability as the receiver rank i approaches n for the outer or combined decoder (or $n + r$ for the inner encoder).

Overall, Fig. 7 illustrates and confirms the packet erasure correction capabilities of DSEP Fulcrum: Since original Fulcrum and DSEP Fulcrum are built on underlying full-vector RLNC [102], where each coded packet equally considers each source data packet, each coded packet in a given generation is capable of "repairing" the loss of any coded packet of the generation. Thus, the number of transmitted extra coded packets for achieving a prescribed decoding success probability is, for the considered one-hop transmission, equivalent to the number packet transmissions that is required to achieve the corresponding number of linear independent coded packets at the receiver over the lossy network. For instance, for $\rho = 1$ extra received coded packet in Fig. 6, $(n+\tau) = (n+\rho)/(1-\epsilon) = 129/0.95 \approx 136$ coded packets, i.e., $\tau = 8$ extra coded packets need to be transmitted in the $\epsilon = 0.05$ case (see Fig. 7). Thus, the curves in Fig. 7 essentially correspond to the curves in Fig. 6 shifted by the factor $1/(1-\epsilon)$ (applied to the total number of received coded packets $n + \rho$) to the right.

We have evaluated the decoding probability with perfect feedback in additional simulations. With perfect feedback, the encoder learns about the impact of a transmitted coded packet on the receiver rank i before encoding the next packet. We found that the decoding probabilities with perfect feedback are very slightly lower, typically only 1–3% lower compared to the decoding probabilities without feedback in Fig. 7. We have furthermore evaluated the decoding throughput for the DSEP policies without feedback and with perfect feedback, see Table 4. The decoding throughput in Table 4 is based on the decoding computation time for a complete generation whereby all coded packets required for decoding are available to the decoder. We observe from Table 4 that the perfect feedback typically increases the decoding throughput by 5–20% compared to the operation without feedback. With perfect feedback, the encoder utilizes the current true receiver rank i for setting the number μ of utilized outer coding expansion packets and the sparsity level $w(i)$ for encoding the next packet. In contrast, without feedback, the encoder assumes that each transmitted coded packet increments the receiver rank i. However, the transmission of a coded packet that is linearly dependent to the already received coded packets or the loss of a transmitted coded packet during network transport

do *not* increment the receiver rank i. Accordingly, without feedback, the encoder tends to overestimate the receiver rank i, i.e., the encoder assumes that the receiver rank i is higher than it actually is. In the DSEP policies, a higher receiver rank i generally leads to a higher number μ of utilized outer coding expansion packets and a higher density w, i.e., to a denser coding with a higher number μ of utilized outer coding expansion packets. The denser coding with a higher number μ of outer coding expansion packets generally increases the decoding probability and increases the decoding computation time, i.e., reduces the decoding throughput.

We further observe from Table 4 that for the DSEP policies without feedback, the decoding throughput somewhat decreases (typically less than 10%) as the packet erasure probability increases from $\epsilon = 0.05$ to 0.1. This is because a higher packet erasure probability ϵ leads to a more pronounced overestimation of the receiver rank i at the encoder as relatively more transmitted packets are lost in the network. Correspondingly, the encoder utilizes relatively more outer expansion packets μ and a higher coding density w, requiring more decoding computations. The DSEP policies with perfect feedback give essentially the same decoding throughput, irrespective of the packet loss probability ϵ, as the perfect feedback continuously gives the encoder the current correct receiver rank i. Throughout, the DSEP inner and combined decoding throughputs are competitive compared to RaptorQ decoding for these smaller values of n.

Overall, we conclude from the evaluations for packet erasures without feedback and with perfect feedback that the practical DSEP policies without feedback continue to perform well for packet erasures: The decoding probability remains high (actually very slightly increases) and the decoding throughput is only somewhat reduced by the packet losses. In particular, for a 10% packet erasure probability, the DSEP policies without feedback still achieve approximately 1.5 fold higher inner and outer decoding throughput, and about 1.4 fold higher combined decoder throughput than the original Fulcrum approach.

4) RECODING IN INTERMEDIATE NETWORK NODES

The DSEP evaluation has so far focused on end-to-end (one-hop) network coding. An important aspect of network coding is recoding in intermediate network nodes [103], [104]. The conventional options for Fulcrum recoding [26] at intermediate nodes tend to cumulatively increase the coding density (decreasing the decoding throughput at the receiver). In particular, for conventional recoding, an intermediate network node buffers all received coded packets for a given generation and XORs a randomly chosen half of the buffered packets with each other to create a recoded packet for onward transmission. (Kodo recoding limits the number of considered coded packets to the most recently received n packets, although more than n coded packets may be received at an intermediate node, e.g., due to packet losses downstream.) As more and more coded packets are received and buffered for a given generation, more and more packets are XORed with each other, which tends to increase the coding density of the resulting recoded packets.

Therefore, DSEP requires novel recoding mechanisms that control the coding density during the recoding. We examine an elementary recoding mechanism for DSEP in this section. This elementary recoding mechanism is suitable for network nodes with limited memory that can store only a few recently received packets. The network node recodes a small prescribed number of the recently received packets e.g., the three most recently received packets, and transmits the recoded packet. This recoding based on a few recently received packets may alter the sparsity from the sparsity at the encoder. However, for scenarios with slowly varying sparsity at the encoder, the deviation from the sparsity level of the encoder should be minor.

One potential problem with recoding a small number of received packets is that all random coding coefficients in $GF(2)$ for linearly combining the received packets (i.e., XORing the received packets with a new coding coefficient of one in $GF(2)$) to create the recoded packet are zero. In order to avoid all zeros, our recoder deterministically assigns the latest (most recent) incoming packet a coding coefficient of one, and randomly generates the remaining coding coefficients with the full density, i.e., each of these remaining coding coefficients is one with probability 1/2. For example, when recoding the three most recently received packets, the possible new coefficients (from oldest to newest of the three packets) can be 0 0 1, 0 1 1, 1 0 1, or 1 1 1 (whereby each of these new coding coefficient vectors occurs with probability 1/4). When beginning the recoding of a generation of coded packets, our elementary DSEP recoding mechanism forwards the first received coded packet of the generation without recoding to the next hop; the second received coded packet is combined with the first received coded packet with probability 0.5, i.e., the new coding coefficients are 0 1 or 1 1 (each with probability 1/2).

In our DSEP evaluation, each intermediate network node always recodes the three most recently received packets to create a recoded packet for transmission, irrespective of whether packet losses have occurred or not. Future recoding refinements could add a protocol mechanism to detect packet erasures, e.g., [105], and to only create a recoded packet when a packet erasure has been detected. To illustrate the effects of the recoding of three buffered packets, consider an example scenario with very sparse encoding with very small $w(i)$ at the beginning of a generation. Suppose that each of the three buffered coded packets is a linear combination of $w(i)$ different source packets, which is likely if $w(i) \ll n$. Our recoding includes the most recently received coded packet with probability one, and each of the two preceding coded packets with probability 1/2. Thus, in the considered scenario, a recoded packet will on average be a linear combination of $2w(i)$ source packets, i.e., the recoding has effectively doubled the coding density.

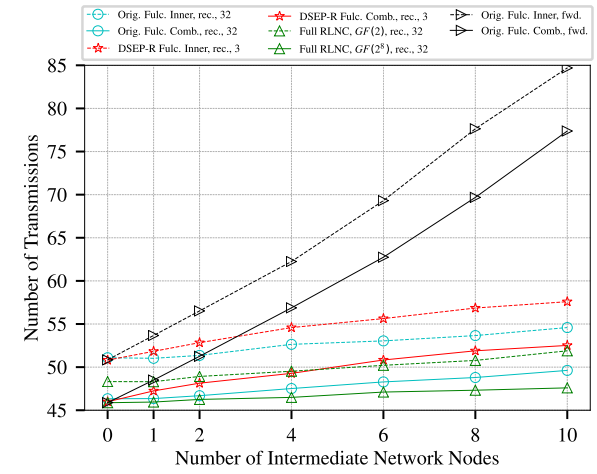
As an extension to the packet loss evaluation in Section V-C3, we consider a linear multi-hop network path

with a prescribed number of intermediate network nodes ranging from 0 (which corresponds to the one-hop scenario in Section V-C3) to 10 intermediate network nodes. The links from the sender to the last intermediate network nodes have a packet erasure probability of $\epsilon = 0.05$, while the link from the last intermediate network node to the receiver has a higher packet erasure probability of $\epsilon = 0.3$ (corresponding to an error-prone wireless last hop).

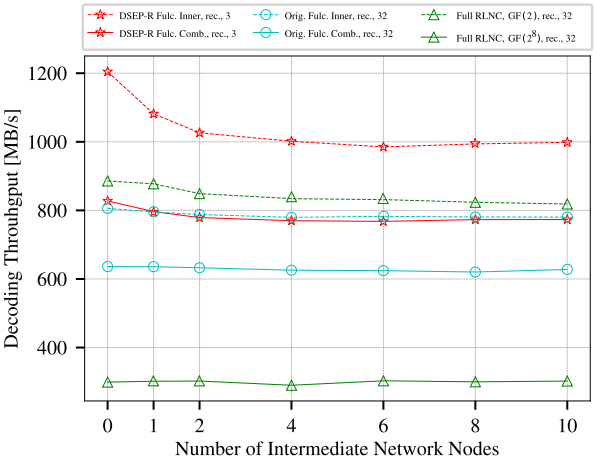
Fig. 8(a) shows the average number of transmitted coded packets that are required for successful decoding as a function of the number of intermediate network nodes. In particular, Fig. 8(a) considers DSEP-R Fulcrum with recoding of the most recently received three coded packets with inner and combined decoding. Furthermore, Fig. 8(a) considers the original Fulcrum with recoding by XORing a randomly chosen half of all (up to n) coded packets that have been received so far in a given generation. For benchmarking, Fig. 8(a) also considers (full-vector) RLNC encoding with full coding density in $GF(2)$ and $GF(2^8)$, with recoding in $GF(2)$ (by XORing) and in $GF(2^8)$ (through linear combination of all buffered packets with new random $GF(2^8)$ coding coefficients), respectively, whereby the recoding considers all buffered received coded packets (i.e., up to n packets when the end of a generation is reached). Moreover, Fig. 8(a) gives the required average number of transmitted coded packets when the intermediate nodes simply forward the original Fulcrum coded packets (without recoding).

We observe from Fig. 8(a) that recoding in intermediate network nodes generally reduces the number of required packet transmissions. The reduction of the number of required packet transmissions becomes more pronounced with increasing number of intermediate network nodes and is due to the inherent gains of in-network recoding [38]. Examining closely the results for zero intermediate network nodes, i.e., a direct link from encoder to decoder with a packet erasure probability of 0.3, we observe that Full RLNC $GF(2)$ requires about 2.3 more transmitted coded packets than Full RLNC $GF(2^8)$. This is due to the linear dependencies of coding a generation of $n = 32$ source packets in $GF(2)$, which requires about 5%, i.e., 1.6, more coded packets for successful decoding than coding in $GF(2^8)$ [95], in combination with the packet loss probability of 0.3, requiring then on average $1.6/0.7 = 2.3$ more coded packet transmissions. We further observe that inner decoding of DSEP-R as well as the original Fulcrum both with recoding and forwarding requires about 51 transmitted coded packets. This is because the design of the inner Fulcrum decoder requires $n + r = 32 + 2$ linearly independent $GF(2)$ coded packets. Considering that approximately 5% additional coded packets need to be received due to linear dependencies, in combination with the 0.3 packet erasure probability, gives on average $34 \cdot 1.05/0.7 = 51$ required coded packet transmissions.

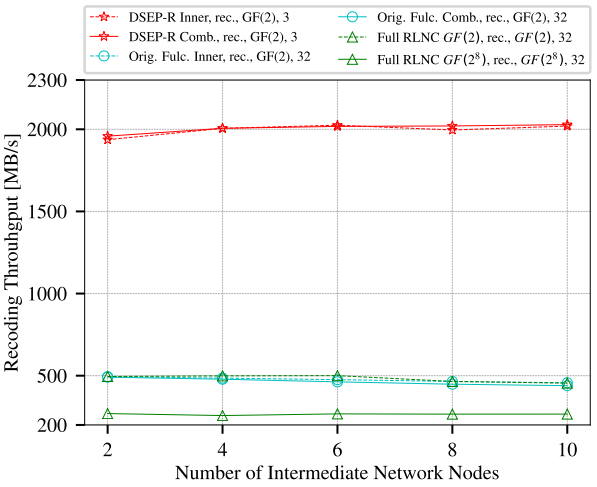
As the number of recoding intermediate network nodes increases, we observe from Fig. 8(a) that original Fulcrum with inner decoding consistently requires about three more transmitted coded packets than RLNC $GF(2)$, while original



(a) # of Transm. Coded Pkts.



(b) Decoding Throughput



(c) Recoding Throughput

FIGURE 8. Evaluation of DSEP recoding with three most recently received packets. Fixed parameters: generation size $n = 32$, $r = 2$ outer coding expansion packets; $\delta = 5$.

Fulcrum with combined decoding tends to require a very slightly increasing number of transmitted coded packets compared to full RLNC $GF(2^8)$. Original Fulcrum employs the

same inner recoding of all buffered received (up to $n = 32$) packets as full RLNC $GF(2)$; however, due to the Fulcrum coding structure with $r = 2$ outer coding expansion packets, the Fulcrum inner decoder requires an extra two linearly independent coded packets at the receiver, which requires on average the transmission of $2 \cdot 1.05/0.7$ more coded packets due to the linear dependence of the $GF(2)$ coding coefficients and the packet erasures. On the other hand, the Fulcrum combined decoder requires the same number of linearly independent coded packets as conventional RLNC decoding, namely n linearly independent coded packets. However, Fulcrum employs only inner recoding in $GF(2)$ in the intermediate network nodes; whereas, full RLNC $GF(2^8)$ recodes in $GF(2^8)$. The inner recoding in $GF(2)$ is computationally less demanding, but is more likely to introduce linear dependencies that require the transmission of additional coded packets.

We observe from Fig. 8(a) that as the number of recoding intermediate network nodes increases, DSEP with inner and combined decoding requires slightly more (at most 2–3 more for 10 intermediate nodes) transmitted coded packets than the corresponding original Fulcrum with inner and combined decoding, respectively. This is mainly due to the sparsity of the recoding in DSEP. Our DSEP recoding considers only the latest three received coded packets; thus, limiting the coding density during the recoding, while incurring more linear dependencies (which in turn implies that more coded packets need to be transmitted to enable decoding). We verified in additional evaluations that considering all (up to $n = 32$) received DSEP coded packets in the recoding would reduce the number of required packet transmissions down to the levels required by the original Fulcrum with recoding of all (up to $n = 32$) received packets. Overall, we observe from Fig. 8(a) that DSEP-R Fulcrum requires only slightly (up to 2–3) more transmitted coded packets than the original Fulcrum across the entire range of examined number of intermediate network nodes and is thus suitable for effective recoding in intermediate network nodes.

Fig. 8(b) shows the decoding throughput as a function of the number of intermediate network nodes where recoding occurred. We observe from Fig. 8(b) that the DSEP-R Fulcrum decoding throughput decreases as the number of intermediate network nodes increases from zero to two. This is mainly due to the increase in coding density resulting from the recoding in the intermediate network nodes. Nevertheless, we observe from Fig. 8(b) that DSEP Fulcrum with inner and combined decoding still achieves substantial decoding throughput increases compared to the respective original Fulcrum schemes and these throughput increases persist as the number of intermediate network nodes increases beyond two. Importantly, DSEP-R with combined decoding, which achieved favorable decoding probabilities in Section III-C3, suffers only minimal decoding throughput reductions with increasing number of intermediate recoding nodes (and correspondingly more lossy transmission hops). With combined decoding, the increase in the inner coding density due to the recoding in the intermediate network nodes causes only a very minor reduction of the decoding throughput since the computational complexity for the combined decoding is dominated by the $GF(2^8)$ computational components of the combined decoding.

Fig 8(c) shows the recoding throughput at an intermediate network node. The recoding throughput is measured as the total recoded data volume, i.e., the number of coded packets times the packet payload size, that has to be processed at the intermediate network nodes to successfully decode a generation at the receiver, divided by the total recoding computation time at all intermediate network nodes. Note that the total recoded data volume is the aggregate of all recoded packets that are created through recoding at all the intermediate network nodes. For instance, the delivery of one coded packet to the receiver via η intermediate network nodes involves a recoded data volume of η data packet payload sizes if there are no packet losses; with $\eta = 1$ intermediate node and a packet loss probability of ϵ_2 on the link from the intermediate node to the receiver, the delivery of one coded packet to the receiver involves on average $1/(1 - \epsilon_2)$ recoded packet payload sizes. The total recoding computation time is the aggregate of the recoding processing times at all intermediate network nodes. Thus, the recoding throughput gives the effective generation rate of recoded packets at a given intermediate network node. We observe from Fig 8(c) that DSEP recoding achieves vastly higher recoding throughput than original Fulcrum. This is mainly due to the limited number of only three packets involved in the DSEP recoding, whereas the original Fulcrum recoding may linearly combine up to $n = 32$ coded packets to create a recoded packet. Full RLNC achieves the same recoding throughput as original Fulcrum as both conduct the same $GF(2)$ recoding operations; while the $GF(2^8)$ recoding of Full RLNC $GF(2^8)$ is computationally more demanding, resulting in a low recoding throughput of just slightly above 200 Mbyte/s. Overall, these recoding throughput results in Fig 8(c) indicate that with DSEP encoding, a network node with the computing capacity of the experimental PC, see Section III-C1, can simultaneously recode two packet traffic flows that operate with the full combined decoding throughput in Fig 8(b). An important future research direction is to further reduce the computational complexity of recoding so that network nodes with limited computational capabilities can recode several traversing packet traffic flows.

An interesting direction for future research is to develop and evaluate novel Fulcrum recoding protocol mechanisms that preserve a prescribed sparse coding density so as to avoid the decrease in decoding throughput observed in Fig. 8(b). A recoding mechanism for intermediate network nodes with abundant memory could store all packets received for a generation thus far and examine the coding coefficients of the received packets to detect their sparsity levels. Then, the intermediate node can judiciously combine selected packets through the XOR operation to approximate a prescribed sparsity level. Note that packet erasures (losses) may make it impossible to achieve exactly the same sparsity level of the

source node, thus some flexible packet selection mechanisms need to be developed to closely approximate the prescribed sparsity level.

VI. CONCLUSION

The recently proposed Fulcrum approach to Random Linear Network Coding (RLNC) combines a high Galois field (GF) outer coding (generating a static number of outer expansion packets) with a static dense small GF inner coding [26]. In the present study, we have generalized Fulcrum coding to dynamically adapt the number of utilized outer expansion packets and the level of density, i.e., equivalently, the sparsity, of the inner coding. In particular, we first examined the four possible combinations of conventional dense coding and static sparse coding for the outer and inner coding in Fulcrum. We concluded that sparse inner with dense outer (SIDO) coding achieves a favorable compromise between high throughput and high decoding probability.

Next, we introduced the dynamic adaptation of the sparsity level as a function of the number of utilized outer expansion packets and the number of linearly independent coded packets at the receiver (i.e., the receiver rank). We then introduced and evaluated dynamic sparsity and expansion packets (DSEP) policies that dynamically adapt the number of utilized outer expansion packets as a function of the receiver rank. We found that the DSEP policies increase the encoding throughput more than tenfold for large generation sizes compared to the conventional Fulcrum approach, while the decoding throughput is increased 1.7 to 4.3 fold. DSEP incurs only very slight reductions (less than 1%) of the decoding probabilities compared to the original Fulcrum approach. We found that the practical DSEP operation without feedback about the receiver rank performs nearly as well as DSEP operation with perfect feedback, even when coded packets are lost during network transport. For 10% packet losses, the feedback-free DSEP still achieved 1.4 fold or higher decoding throughput increases compared to the original Fulcrum approach.

The DSEP concept that we introduced in this article and the corresponding publicly available code at https://github.com/nguyenvutud/DSEP open up several interesting direction for future research. The present study has focused on generation based RLNC. It would be interesting in future research to explore the DSEP concept in the context of sliding window RLNC [42]–[47]. For sliding window RLNC, a DSEP policy could adapt the sparsity and extra coded packets according to the status of the receiver for a given current sliding window position. Furthermore, the DSEP Fulcrum coding in this article has been limited to non-systematic coding (specifically, only the outer coding has been systematic, but the inner coding has been non-systematic); an important future research direction is to extend DSEP Fulcrum coding to fully systematic coding [15], [33]–[35] so as to allow the immediate transmission of the uncoded source packets without any encoding delay. Additionally, the speed-up of the DSEP Fulcrum encoding and decoding computations through specialized parallel computing strategies on multicore platforms [16]–[19] should be explored in future research. Moreover, the sparsity and extra coded packets could be adapted according to the loss conditions, e.g., high-loss periods could delay the increase of the receiver rank. The present study has examined the DSEP concept in the context of the Fulcrum multi-layer networking coding paradigm. Future research could explore the DSEP concept in other multi-layer coding paradigms involving network coding, e.g., in the context of batched sparse (BATS) network coding based on generations [106], [107] or a sliding window [94], employing RLNC as the inner code and a Luby transform outer code [52].

ACKNOWLEDGMENT

The authors are grateful to Michael Luby and his team at the International Computer Science Institute (ICSI), Berkeley, CA, USA, for sharing the CodornicesRq (Rel. 2.1) evaluation implementation of the RaptorQ code, and helping to understand how to interpret some of the evaluation results. A preliminary version of static sparse Fulcrum coding appeared in [1], while a preliminary version of dynamic Fulcrum expansion packets (without sparsity) appeared in [2].

REFERENCES

[1] V. Nguyen, G. T. Nguyen, F. Gabriel, D. E. Lucani, and F. H. P. Fitzek, "Integrating sparsity into Fulcrum codes: Investigating throughput, complexity and overhead," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2018, pp. 1–6.

[2] V. Nguyen, E. Tasdemir, G. Nguyen, D. Lucani, and F. Fitzek, "Tunable expansion packets for Fulcrum codes," in *Proc. Eur. Wireless*, 2019, pp. 1–7.

[3] H. Alshaheen and H. Takruri-Rizk, "Energy saving and reliability for wireless body sensor networks (WBSN)," *IEEE Access*, vol. 6, pp. 16678–16695, 2018.

[4] Y. N. Shnaiwer, S. Sorour, T. Y. Al-Naffouri, and S. N. Al-Ghadhban, "Opportunistic network coding-assisted cloud offloading in heterogeneous fog radio access networks," *IEEE Access*, vol. 7, pp. 56147–56162, 2019.

[5] R. Torrea-Duran, M. Morales Cespedes, J. Plata-Chaves, L. Vandendorpe, and M. Moonen, "Topology-aware space-time network coding in cellular networks," *IEEE Access*, vol. 6, pp. 7565–7578, 2018.

[6] H. Kang, H. Yoo, D. Kim, and Y.-S. Chung, "CANCORE: Context-aware network COded REpetition for VANETs," *IEEE Access*, vol. 5, pp. 3504–3512, 2017.

[7] X. Shao, C. Wang, C. Zhao, and J. Gao, "Traffic shaped network coding aware routing for wireless sensor networks," *IEEE Access*, vol. 6, pp. 71767–71782, 2018.

[8] J.-W. Kim and J.-S. No, "Code equivalences between network codes with link errors and index codes with side information errors," *IEEE Access*, vol. 7, pp. 54144–54154, 2019.

[9] F. A. Monteiro, A. Burr, I. Chatzigeorgiou, C. Hollanti, I. Krikidis, H. Seferoglu, and V. Skachek, "Special issue on network coding," *EURASIP J. Adv. Signal Process.*, vol. 2017, no. 1, p. 29, Dec. 2017.

[10] H. V. Nguyen, S. X. Ng, W. Liang, P. Xiao, and L. Hanzo, "A network-coding aided road-map of large-scale near-capacity cooperative communications," *IEEE Access*, vol. 6, pp. 21592–21620, 2018.

[11] Q. Wang, X. Zhang, Q. Wang, P. Liu, and B. Deng, "The network coding algorithm based on rate selection for device-to-device communications," *IEEE Access*, vol. 7, pp. 23396–23406, 2019.

[12] C. Zhang, C. Li, and Y. Chen, "Joint opportunistic routing and intra-flow network coding in multi-hop wireless networks: A survey," *IEEE Netw.*, vol. 33, no. 1, pp. 113–119, Jan. 2019.

[13] A. Ahmed, H. Shan, and A. Huang, "Modeling the delivery of coded packets in D2D mobile caching networks," *IEEE Access*, vol. 7, pp. 20091–20105, 2019.

[14] W. He, Y. Su, X. Xu, Z. Luo, L. Huang, and X. Du, "Cooperative content caching for mobile edge computing with network coding," *IEEE Access*, vol. 7, pp. 67695–67707, 2019.

[15] N. Ma and M. Diao, "CoFi: Coding-assisted file distribution over a wireless LAN," *Symmetry*, vol. 11, no. 1, p. 71, 2019.
[16] H. Shin and J.-S. Park, "Optimizing random network coding for multimedia content distribution over smartphones," *Multimedia Tools Appl.*, vol. 76, no. 19, pp. 19379–19395, Oct. 2017.
[17] H. Shin and J.-S. Park, "Reducing energy consumption of RNC based media streaming on smartphones via sampling," *Multimedia Tools Appl.*, vol. 78, no. 20, pp. 28461–28475, Oct. 2019.
[18] S. Wunderlich, J. A. Cabrera, F. H. P. Fitzek, and M. Reisslein, "Network coding in heterogeneous multicore IoT nodes with DAG scheduling of parallel matrix block operations," *IEEE Internet Things J.*, vol. 4, no. 4, pp. 917–933, Aug. 2017.
[19] S. Wunderlich, F. H. P. Fitzek, and M. Reisslein, "Progressive multicore RLNC decoding with online DAG scheduling," *IEEE Access*, vol. 7, pp. 161184–161200, 2019.
[20] A. J. Ferrer, J. M. Marques, and J. Jorba, "Towards the decentralised cloud: Survey on approaches and challenges for mobile, ad hoc, and edge computing," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 111.1–111.36, Feb. 2019.
[21] M. Mehrabi, D. You, V. Latzko, H. Salah, M. Reisslein, and F. H. P. Fitzek, "Device-enhanced MEC: Multi-access edge computing (MEC) aided by end device computation and caching: A survey," *IEEE Access*, vol. 7, pp. 166079–166108, 2019.
[22] D. Chen, J. Cong, S. Gurumani, W.-M. Hwu, K. Rupnow, and Z. Zhang, "Platform choices and design demands for IoT platforms: Cost, power, and performance tradeoffs," *IET Cyber-Phys. Syst., Theory Appl.*, vol. 1, no. 1, pp. 70–77, Dec. 2016.
[23] S. J. Johnston, M. Apetroaie-Cristea, M. Scott, and S. J. Cox, "Applicability of commodity, low cost, single board computers for Internet of Things devices," in *Proc. IEEE 3rd World Forum Internet Things (WF-IoT)*, Dec. 2016, pp. 1–6.
[24] S. J. Johnston, P. J. Basford, C. S. Perkins, H. Herry, F. P. Tso, D. Pezaros, R. D. Mullins, E. Yoneki, S. J. Cox, and J. Singer, "Commodity single board computer clusters and their applications," *Future Gener. Comput. Syst.*, vol. 89, pp. 201–212, Dec. 2018.
[25] A. Raza, A. A. Ikram, A. Amin, and A. J. Ikram, "A review of low cost and power efficient development boards for IoT applications," in *Proc. Future Technol. Conf. (FTC)*, Dec. 2016, pp. 786–790.
[26] D. E. Lucani, M. V. Pedersen, D. Ruano, C. W. Sorensen, F. H. P. Fitzek, J. Heide, O. Geil, V. Nguyen, and M. Reisslein, "Fulcrum: Flexible network coding for heterogeneous devices," *IEEE Access*, vol. 6, pp. 77890–77910, 2018.
[27] S. Brown, O. Johnson, and A. Tassi, "Reliability of broadcast communications under sparse random linear network coding," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4677–4682, May 2018.
[28] Y. Li, J. Wang, S. Zhang, Z. Bao, and J. Wang, "Efficient coastal communications with sparse network coding," *IEEE Netw.*, vol. 32, no. 4, pp. 122–128, Jul. 2018.
[29] Y. Li, E. Soljanin, and P. Spasojevic, "Effects of the generation size and overlap on throughput and complexity in randomized linear network coding," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 1111–1123, Feb. 2011.
[30] P. Maymounkov, N. J. Harvey, and D. S. Lun, "Methods for efficient network coding," *Proc. Allerton Conf. Commun., Control, Comp.*, 2006, pp. 482–491.
[31] C. W. Sorensen, A. S. Badr, J. A. Cabrera, D. E. Lucani, J. Heide, and F. H. Fitzek, "A practical view on tunable sparse network coding," *Proc. VDE Eur. Wireless Conf.*, 2015, pp. 1–6.
[32] A. Tassi, I. Chatzigeorgiou, and D. E. Lucani, "Analysis and optimization of sparse random linear network coding for reliable multicast services," *IEEE Trans. Commun.*, vol. 64, no. 1, pp. 285–299, Jan. 2016.
[33] J. Heide, M. V. Pedersen, F. H. P. Fitzek, and T. Larsen, "Network coding for mobile Devices–Systematic binary random rateless codes," in *Proc. IEEE Int. Conf. Commun. Workshops*, Jun. 2009, pp. 1–6.
[34] Y. Li, S. Blostein, and W.-Y. Chan, "Systematic network coding for two-hop lossy transmissions," *EURASIP J. Adv. Signal Process.*, vol. 2015, no. 1, pp. 1–14, Dec. 2015.
[35] S. Pandi, F. Gabriel, J. A. Cabrera, S. Wunderlich, M. Reisslein, and F. H. P. Fitzek, "PACE: Redundancy engineering in RLNC for low-latency communication," *IEEE Access*, vol. 5, pp. 20477–20493, 2017.
[36] V. Nguyen, J. A. Cabrera, G. T. Nguyen, F. Gabriel, C. Lehmann, S. Mudriievskyi, and F. H. P. Fitzek, "Adaptive decoding for Fulcrum codes," in *Proc. IEEE 9th Annu. Inf. Technol., Electron. Mobile Commun. Conf. (IEMCON)*, Nov. 2018, pp. 133–139.
[37] V. Nguyen, J. A. Cabrera, D. You, H. Salah, G. T. Nguyen, and F. H. P. Fitzek, "Advanced adaptive decoder using Fulcrum network codes," *IEEE Access*, vol. 7, pp. 141648–141661, 2019.
[38] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.
[39] J. He, V. Tervo, X. Zhou, X. He, S. Qian, M. Cheng, M. Juntti, and T. Matsumoto, "A tutorial on lossy forwarding cooperative relaying," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 66–87, 1st Quart., 2019.
[40] L. Wei and W. Chen, "Compute-and-forward network coding design over multi-source multi-relay channels," *IEEE Trans. Wireless Commun.*, vol. 11, no. 9, pp. 3348–3357, Sep. 2012.
[41] M. El Soussi, A. Zaidi, and L. Vandendorpe, "Compute-and-forward on a multiaccess relay channel: Coding and symmetric-rate optimization," *IEEE Trans. Wireless Commun.*, vol. 13, no. 4, pp. 1932–1947, Apr. 2014.
[42] F. Gabriel, S. Wunderlich, S. Pandi, F. H. P. Fitzek, and M. Reisslein, "Caterpillar RLNC with feedback (CRLNC-FB): Reducing delay in selective repeat ARQ through coding," *IEEE Access*, vol. 6, pp. 44787–44802, 2018.
[43] D. Malak, E. Ohad, M. Médard, and E. M. Yeh, "Throughput and delay analysis for coded ARQ," *Proc. IFIP Int. Symp. Model. Optim. Mobile, Ad Hoc, Wireless Netw. (WIOPT)*, 2019, pp. 1–8.
[44] D. Malak, M. Medard, and E. M. Yeh, "Tiny codes for guaranteeable delay," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 4, pp. 809–825, Apr. 2019.
[45] J. K. Sundararajan, D. Shah, M. Medard, and P. Sadeghi, "Feedback-based online network coding," *IEEE Trans. Inf. Theory*, vol. 63, no. 10, pp. 6628–6649, Oct. 2017.
[46] P. U. Tournoux, E. Lochin, J. Lacan, A. Bouabdallah, and V. Roca, "On-the-Fly erasure coding for real-time video applications," *IEEE Trans. Multimedia*, vol. 13, no. 4, pp. 797–812, Aug. 2011.
[47] 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. 20183–20197, 2017.
[48] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 4, pp. 56–67, Oct. 1998.
[49] J. W. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1528–1540, Oct. 2002.
[50] J. W. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain retrospective," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 49, no. 5, pp. 82–85, Nov. 2019.
[51] D. J. C. MacKay, "Fountain codes," *IEE Proc. Commun.*, vol. 152, no. 6, pp. 1062–1068, 2005.
[52] M. Luby, "LT codes," in *Proc. 43rd Annu. IEEE Symp. Found. Comput. Sci.*, Nov. 2002, pp. 271–280.
[53] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer, *Raptor Forward Error Correction Scheme for Object Delivery*, document RFC 5053, Oct. 2007. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5053.txt
[54] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.
[55] A. Shokrollahi and M. Luby, "Raptor codes," *Found. Trends Commun. Inf. Theory*, vol. 6, nos. 3–4, pp. 213–322, May 2011.
[56] C. Bouras, N. Kanakis, V. Kokkinos, and A. Papazois, "Embracing RaptorQ FEC in 3GPP multicast services," *Wireless Netw.*, vol. 19, no. 5, pp. 1023–1035, Jul. 2013.
[57] S. Puducheri, J. Kliewer, and T. E. Fuja, "The design and performance of distributed LT codes," *IEEE Trans. Inf. Theory*, vol. 53, no. 10, pp. 3740–3754, Oct. 2007.
[58] J. He, I. Hussain, Y. Li, M. Juntti, and T. Matsumoto, "Distributed LT codes with improved error floor performance," *IEEE Access*, vol. 7, pp. 8102–8110, 2019.
[59] Qualcomm. (2010) *RaptorQ–Technical Overview*. Accessed: Oct. 27, 2018. [Online]. Available: http://www.qualcomm.com/media/documents/raptorq-technical-overview.pdf
[60] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder, *RaptorQ Forward Error Correction Scheme for Object Delivery*, document RFC 6330, Aug. 2011. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6330.txt
[61] K. Zhang, Q. Zhang, and J. Jiao, "Bounds on the reliability of RaptorQ codes in the finite-length regime," *IEEE Access*, vol. 5, pp. 24766–24774, 2017.
[62] A. Ali, K. S. Kwak, N. H. Tran, Z. Han, D. Niyato, F. Zeshan, M. T. Gul, and D. Y. Suh, "RaptorQ-based efficient multimedia transmission over cooperative cellular cognitive radio networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 8, pp. 7275–7289, Aug. 2018.

[63] H. Chen, X. Zhang, Y. Xu, Z. Ma, and W. Zhang, "Efficient mobile video streaming via context-aware RaptorQ-based unequal error protection," *IEEE Trans. Multimedia*, vol. 22, no. 2, pp. 459–473, Feb. 2020.

[64] M. Talha Gul, A. Ali, D. K. Singh, U. Imtinan, I. Raza, S. A. Hussain, D. Y. Suh, and J.-W. Lee, "Merge-and-forward: A cooperative multimedia transmissions protocol using RaptorQ codes," *IET Commun.*, vol. 10, no. 15, pp. 1884–1895, Oct. 2016.

[65] M. Taghouti, D. E. Lucani, J. A. Cabrera, M. Reisslein, M. V. Pedersen, and F. H. P. Fitzek, "Reduction of padding overhead for RLNC media distribution with variable size packets," *IEEE Trans. Broadcast.*, vol. 65, no. 3, pp. 558–576, Sep. 2019.

[66] N. Thomos and P. Frossard, "Toward one symbol network coding vectors," *IEEE Commun. Lett.*, vol. 16, no. 11, pp. 1860–1863, Nov. 2012.

[67] M. Luby and L. Minder. (Dec. 2019). *How to use the Codornices Rq Software Package, Version v2.1*. Accessed: Jan. 17, 2020. [Online]. Available: http://www.codornices.info

[68] I. Chatzigeorgiou and A. Tassi, "Decoding delay performance of random linear network coding for broadcast," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7050–7060, Aug. 2017.

[69] A. Douik and S. Sorour, "Data dissemination using instantly decodable binary codes in fog-radio access networks," *IEEE Trans. Commun.*, vol. 66, no. 5, pp. 2052–2064, May 2018.

[70] M. Nistor, D. E. Lucani, T. T. V. Vinhoza, R. A. Costa, and J. Barros, "On the delay distribution of random linear network coding," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 5, pp. 1084–1093, May 2011.

[71] J. Qureshi, C. H. Foh, and J. Cai, "Online XOR packet coding: Efficient single-hop wireless multicasting with low decoding delay," *Comput. Commun.*, vol. 39, pp. 65–77, Feb. 2014.

[72] H. Tang, Q. T. Sun, Z. Li, X. Yang, and K. Long, "Circular-shift linear network coding," *IEEE Trans. Inf. Theory*, vol. 65, no. 1, pp. 65–80, Jan. 2019.

[73] S. Feizi, D. E. Lucani, and M. Médard, "Tunable sparse network coding," in *Proc. Int. Zurich Seminar Commun. (IZS)*, Zürich, Switzerland, Feb. 2012, pp. 107–110.

[74] A. Fiandrotti, V. Bioglio, M. Grangetto, R. Gaeta, and E. Magli, "Band codes for energy-efficient network coding with application to P2P mobile streaming," *IEEE Trans. Multimedia*, vol. 16, no. 2, pp. 521–532, Feb. 2014.

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

[76] Y. Li, W.-Y. Chan, and S. D. Blostein, "On design and efficient decoding of sparse random linear network codes," *IEEE Access*, vol. 5, pp. 17031–17044, 2017.

[77] B. Tang, S. Yang, B. Ye, Y. Yin, and S. Lu, "Expander chunked codes," *EURASIP J. Adv. Signal Process.*, vol. 2015, no. 1, pp. 1–13, Dec. 2015.

[78] S. Yang and R. W. Yeung, "Batched sparse codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 9, pp. 5322–5346, Sep. 2014.

[79] H. Y. Kwan, K. W. Shum, and C. W. Sung, "Generation of innovative and sparse encoding vectors for broadcast systems with feedback," in *Proc. IEEE Int. Symp. Inf. Theory Proc.*, Jul. 2011, pp. 1161–1165.

[80] C. W. Sung, K. W. Shum, and H. Y. Kwan, "On the sparsity of a linear network code for broadcast systems with feedback," in *Proc. Int. Symp. Netw. Coding*, Jul. 2011, pp. 1–4.

[81] C. W. Sung, K. W. Shum, L. Huang, and H. Y. Kwan, "Linear network coding for erasure broadcast channel with feedback: Complexity and algorithms," *IEEE Trans. Inf. Theory*, vol. 62, no. 5, pp. 2493–2503, May 2016.

[82] W. Li, F. Bassi, and M. Kieffer, "Sparse random linear network coding for data compression in WSNs," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 2729–2733.

[83] A. Tassi, R. J. Piechocki, and A. Nix, "On intercept probability minimization under sparse random linear network coding," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 6137–6141, Jun. 2019.

[84] I. Chatzigeorgiou, G. Kurt, S. T. Basaran, and A. S. Khan, "On the decoding failure probability of random network coded cooperation," in *Proc. IEEE 89th Veh. Technol. Conf. (VTC-Spring)*, Apr. 2019, pp. 1–5.

[85] G. Gankhuyag, E. Hong, and Y. Choe, "Sparse recovery using sparse sensing matrix based finite field optimization in network coding," *IEICE Trans. Inf. Syst.*, vol. E100.D, no. 2, pp. 375–378, 2017.

[86] P. Garrido, D. E. Lucani, and R. Aguero, "Markov chain model for the decoding probability of sparse network coding," *IEEE Trans. Commun.*, vol. 65, no. 4, pp. 1675–1685, Apr. 2017.

[87] H. Sehat and P. Pahlevani, "An analytical model for rank distribution in sparse network coding," *IEEE Commun. Lett.*, vol. 23, no. 4, pp. 556–559, Apr. 2019.

[88] A. Zarei, P. Pahlevani, and M. Davoodi, "On the partial decoding delay of sparse network coding," *IEEE Commun. Lett.*, vol. 22, no. 8, pp. 1668–1671, Aug. 2018.

[89] S. Feizi, D. E. Lucani, C. W. Sorensen, A. Makhdoumi, and M. Medard, "Tunable sparse network coding for multicast networks," in *Proc. Int. Symp. Netw. Coding (NetCod)*, Jun. 2014, pp. 1–6.

[90] P. Garrido, C. W. Sorensen, D. E. Lucani, and R. Aguero, "Performance and complexity of tunable sparse network coding with gradual growing tuning functions over wireless networks," in *Proc. IEEE 27th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Sep. 2016, pp. 1–6.

[91] P. Garrido, D. E. Lucani, and R. Aguero, "How to tune sparse network coding over wireless links," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar. 2017, pp. 1–6.

[92] C. W. Sorensen, D. E. Lucani, F. H. P. Fitzek, and M. Medard, "On-the-Fly overlapping of sparse generations: A tunable sparse network coding perspective," in *Proc. IEEE 80th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2014, pp. 1–5.

[93] P. Garrido, D. Gomez, J. Lanza, and R. Aguero, "Exploiting sparse coding: A sliding window enhancement of a random linear network coding scheme," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.

[94] J. Yang, Z.-P. Shi, C.-X. Wang, and J.-B. Ji, "Design of optimized sliding-window BATS codes," *IEEE Commun. Lett.*, vol. 23, no. 3, pp. 410–413, Mar. 2019.

[95] J. Heide, M. V. Pedersen, F. H. P. Fitzek, and M. Medard, "On code parameters and coding vector representation for practical RLNC," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2011, pp. 1–5.

[96] M. V. Pedersen, J. Heide, and F. Fitzek, "Kodo: An open and research oriented network coding library," *Lect. Notes Comput. Sci.*, vol. 6827, pp. 145–152, Dec. 2011.

[97] J. Heide, M. V. Pedersen, and F. H. Fitzek, "Decoding algorithms for random linear network codes," in *Proc. Int. Conf. Res. Netw.* Berlin, Germany: Springer, 2011, pp. 129–136.

[98] C. W. Sorensen, A. Paramanathan, J. A. Cabrera, M. V. Pedersen, D. E. Lucani, and F. H. P. Fitzek, "Leaner and meaner: Network coding in SIMD enabled commercial devices," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Apr. 2016, pp. 1–6.

[99] N. H. Marcano, C. Sorensen, J. Cabrera, G. S. Wunderlich, D. Lucani, and F. Fitzek, "On goodput and energy measurements of network coding schemes in the raspberry pi," *Electronics*, vol. 5, no. 4, p. 66, 2016.

[100] M. Luby and L. Minder. (May 2019). *Performance of Codornices Rq Software Package*. Accessed: Jan. 2, 2020. [Online]. Available: http://www1.icsi.berkeley.edu/~pooja/PerformanceCodornicesRqReleas

[101] Steinwurf ApS. (Sep. 2019), *Kodo Throughput Benchmarks*. Accessed: Jan. 2, 2020. [Online]. Available: https://www.steinwurf.com/assets/images/blog/Kodo_Throughput_Benchmarks_Comparison_RaptorQ.pdf

[102] 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. Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[103] P. Garrido, D. Lucani, and R. Aguero, "Role of intermediate nodes in sparse network coding: Characterization and practical recoding," in *Proc. Eur. Wireless Conf.*, May 2017, pp. 1–7.

[104] P. Garrido, A. Fernandez, and R. Aguero, "To recode or not to recode: Optimizing RLNC recoding and performance evaluation over a COTS platform," in *Proc. Eur. Wireless Conf.*, May 2018, pp. 1–7.

[105] P. Pahlevani, D. E. Lucani, M. V. Pedersen, and F. H. P. Fitzek, "PlayNCool: Opportunistic network coding for local optimization of routing in wireless mesh networks," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2013, pp. 812–817.

[106] X. Xu, Y. L. Guan, Y. Zeng, and C.-C. Chui, "Quasi-universal BATS code," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3497–3501, Apr. 2017.

[107] S. Yang and R. W. Yeung, *BATS Codes: Theory and Practice*. San Rafael, CA, USA: Morgan & Claypool, 2017.

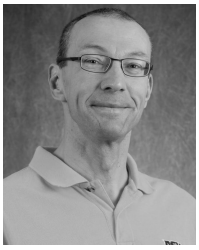
VU NGUYEN (Graduate Student Member, IEEE) received the B.Tech. (IT) degree from the Hue University of Education, Vietnam, in 2006, and the M.Eng. degree in computer science from the Da Nang University of Technology, Vietnam, in 2011. He was a Lecturer with the Vietnam-Korea Friendship IT College, Da Nang, Vietnam. He is currently pursuing the Ph.D. degree with the Deutsch Telekom Chair of Communication Networks, Dresden University of Technology, Germany. He is interested in the research areas of network coding and high performance and low complexity communications.

ELIF TASDEMIR received the B.Sc. degree in electronics and telecommunication engineering from Kocaeli University, in 2012, and the M.Sc. degree in communication engineering from Yildiz Technical University, Turkey, in 2015. She is currently pursuing the Ph.D. degree with the Deutsche Telekom Chair of Communication Networks, TU Dresden.

GIANG T. NGUYEN (Member, IEEE) received the M.Eng. degree in telecommunications from the Asian Institute of Technology (AIT), Thailand, in 2007, and the Ph.D. degree (Dr. Ing.) in computer science from TU Dresden, Germany, in 2016. He is currently a Senior Researcher with the 5G Lab Germany. His research interests include the area of network function virtualization (NFV) and mobile edge computing (MEC) for 5G networks, the reliability aspects of peer-to-peer video streaming, network coding, and low-latency networking.

DANIEL E. LUCANI (Senior Member, IEEE) received the B.S. *(summa cum laude)* and M.S. (Hons.) degrees in electronics engineering from Universidad Simón Bolívar, Caracas, Venezuela, in 2005 and 2006, respectively, and the Ph.D. degree in electrical engineering from the Massachusetts Institute of Technology (MIT), in 2010. He was an Associate Professor with Aalborg University, from 2012 to 2017, and an Assistant Professor with the University of Porto, from 2010 to 2012. He has been an Associate Professor with the Department of Engineering, Aarhus University, since April 2017, and the CEO and the Lead Scientist of the start-up company Chocolate Cloud ApS, since June 2014. He has published more than 140 scientific articles in international journals and top-ranked international conferences and eight patents and patent applications. His research interests include communications and networks, network coding, information theory, coding theory, distributed storage and computation, and their applications to cloud computing technologies necessary to enable the Internet of Things (IoT), big data, and 5G applications and services. He was a recipient of the IEEE ComSoc Outstanding Young Researcher Award for the EMEA Region, in 2015, and the Danish Free Research Foundation's Sapere Aude Starting Grant. He was the General Co-Chair of the 2014 International Symposium on Network Coding (NetCod2014). He is an Associate Editor of the *EURASIP Journal on Wireless Communications and Networking*.

FRANK H. P. FITZEK (Senior Member, IEEE) received the diploma (Dipl.Ing.) degree in electrical engineering from the University of Technology–Rheinisch-Westfälische Technische Hochschule (RWTH), Aachen, Germany, in 1997, and the Ph.D. (Dr. Ing.) degree in electrical engineering from the Technical University of Berlin, Germany, in 2002. He is currently a Professor and the Head of the Deutsche Telekom Chair of Communication Networks, Technical University Dresden, Germany, coordinating the 5G Lab Germany. He is the spokesman of the DFG Cluster of Excellence CeTI. He became an Adjunct Professor with the University of Ferrara, Italy, in 2002. In 2003, he joined Aalborg University as an Associate Professor and later became a Professor. He co-founded several start-up companies starting with Acticom GmbH, Berlin, in 1999. His current research interests are in the areas of wireless and mobile 5G communication networks, mobile phone programming, network coding, cross layer as well as energy efficient protocol design, and cooperative networking. He was selected to receive the NOKIA Champion Award several times in a row, from 2007 to 2011. In 2008, he was awarded the Nokia Achievement Award for his work on cooperative networks. In 2011, he received the SAPERE AUDE Research Grant from the Danish Government, the Vodafone Innovation Prize, in 2012. In 2015, he was awarded the Honorary Degree (Doctor Honoris Causa) from the Budapest University of Technology and Economy (BUTE).



MARTIN REISSLEIN (Fellow, IEEE) received the Ph.D. degree in systems engineering from the University of Pennsylvania, in 1998. He is currently a Professor with the School of Electrical, Computer, and Energy Engineering, Arizona State University (ASU), Tempe, and an external associated Investigator with the Centre for Tactile Internet With Human-in-the-Loop (CeTI), Technische Universität Dresden, Germany. He received the IEEE Communications Society Best Tutorial Paper Award, in 2008, the Friedrich Wilhelm Bessel Research Award from the Alexander von Humboldt Foundation, in 2015, and the Dresden Senior Fellowship, in 2016 and 2019. He is an Associate Editor-in-Chief of the IEEE COMMUNICATIONS SURVEYS & TUTORIALS, a Co-Editor-in-Chief of *Optical Switching and Networking*, and chaired the Steering Committee of the IEEE TRANSACTIONS ON MULTIMEDIA, from 2017 to 2019. He currently serves as an Associate Editor for the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON EDUCATION, the IEEE ACCESS, and *Computer Networks*.