IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 10, 2021, accepted December 17, 2021, date of publication December 21, 2021,
date of current version December 30, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3137209

SpaRec: Sparse Systematic RLNC Recoding in Multi-Hop Networks

ELIF TASDEMIR[1], MÁTÉ TÖMÖSKÖZI[1,2], JUAN A. CABRERA[1], FRANK GABRIEL[1], DONGHO YOU[3], FRANK H. P. FITZEK[1,2], (Senior Member, IEEE), AND MARTIN REISSLEIN[4], (Fellow, IEEE)

[1]5G Lab Germany, Deutsche Telekom Chair, Technische Universität Dresden, 01062 Dresden, Germany
[2]Centre for Tactile Internet With Human-in-the-Loop (CeTI), Technische Universität Dresden, 01062 Dresden, Germany
[3]Department of Information and Communication Engineering, Hannam University, Daejeon 34430, Republic of Korea
[4]School of Electrical, Computing, and Energy Engineering, Arizona State University, Tempe, AZ 85287, 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—Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop" (CeTI) of Technische Universität Dresden under Project 390696704, and in part by the Ministry of National Education Turkey.

ABSTRACT Sparse Random Linear Network Coding (RLNC) reduces the computational complexity of the RLNC decoding through a low density of the non-zero coding coefficients, which can be achieved through sending uncoded (systematic) packets. However, conventional recoding of sparse RLNC coded packets at an intermediate node in a multi-hop network increases the density of non-zero coding coefficients. We develop and evaluate sparsity-preserving recoding (SpaRec) strategies that preserve the low density of non-zero coding coefficients of sparse RLNC with systematic packets. We develop SpaRec strategies with and without decoding at the intermediate nodes, with and without a specified coding rate, as well as with finite and infinite recoding window lengths. We evaluate the SpaRec strategies in multi-hop networks in terms of packet loss, packet delivery delay, as well as recoding and decoding (computation) throughput. We find that the SpaRec strategies substantially improve the RLNC performance compared to conventional recoding.

INDEX TERMS Intermediate network node, low coding coefficient density, multi-hop network, random linear network coding (RLNC), recoding, sliding window coding, sparse coding, systematic packet.

I. INTRODUCTION

A. MOTIVATION

Random Linear Network Coding (RLNC) linearly combines a block (also referred to as generation) of G data packets with random coding coefficients from a Galois Field to create coded packets. The original G data packets can be recovered from any set of G received linearly independent coded packets through RLNC decoding. The decoding involves the multiplication and inversion of a $G \times G$ matrix of coding coefficients. The RLNC packet recovery from any set of G linearly independent coded packets does not require any signaling or coordination. Each coded packet only needs to carry its own coding coefficients, making RLNC well-suited for error-prone communication networks, e.g., wireless networks [3]–[8] and content distribution networks [9]–[11]. However, the high computational complexity of the RLNC coding and decoding has hampered the adoption of RLNC in practical networks [12], [13].

One promising strategy to reduce the computational complexity while retaining the packet recovery property is to employ sparse coding coefficients, i.e., a low density of non-zero coding coefficients [14]. As reviewed in Section I-B, a low-coding coefficient density can be achieved with various strategies [14]. We focus on the systematic RLNC approach that intersperses only few coded packets with non-zero coding coefficients among the uncoded (so-called systematic) packets in a generation, i.e., so-called *sparse systematic RLNC* [15]–[19]. To date, sparse systematic RLNC has been mainly studied for end-to-end coding, i.e., sparse systematic RLNC encoding at the sending (source) node, and decoding at the ultimate destination node. However, an important performance-enhancing aspect of RLNC is recoding at intermediate network nodes, which can substantially increase the overall network performance [20]–[29]. With recoding, the intermediate network nodes not only store-and-forward

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

coded packets, but rather store, recode, and forward the packets.

Conventional recoding typically combines the received packets with new random coding coefficients without consideration of the systematic RLNC coding structure of systematic packets and coded packets. Thereby, the conventional recoding effectively "destroys" the sparsity property of the initially sparse systematic RLNC encoding of the source.

B. RELATED WORK

Pioneering studies on sparse RLNC, e.g., [30]–[34], primarily considered sparse RLNC with random subsets of the coding coefficients set to zero. Refinements and variations of this random subset sparsity approach have been investigated in numerous studies, e.g. [35]–[45], and related recoding strategies have been studied in [46], [47]. Li *et al.* [14] have given a general overview of sparse network coding schemes and quantitatively compared the sparse systematic RLNC with batched sparse (BATS) network coding [48]–[52]. Li *et al.* have conducted the quantitative comparison in the context of coastal communication with recoding in one intermediate network node and found that BATS coding and systematic RLNC have generally comparable performance; whereby, BATS coding tends to achieve slightly higher data rates and overall lower decoding complexity, whereas systematic RLNC tends to achieve shorter delays until decoding the full generation at the destination as well as lower decoding complexity for networks with low packet loss probabilities.

While recoding in intermediate network nodes has been extensively studied for BATS codes, see e.g., [53]–[55], to the best of our knowledge there has been no prior detailed study of *recoding* for sparse systematic RLNC with preservation of the sparsity property. This journal article extends the related preliminary conference papers [1], [2] by introducing the novel Rec-woDec strategy and the novel rateless recoding approach. Also, this journal article provides substantially extended evaluations compared to [1], [2] by including the recoding throughput as well as evaluations for different numbers of intermediate nodes.

Systematic coding is generally preferred for low-latency communication [56]–[59] and there have been extensive studies on low-latency communication schemes employing systematic RLNC, see e.g., [60]–[66]. Therefore, it is important to develop and evaluate sparsity-preserving recoding strategies for sparse systematic RLNC.

C. CONTRIBUTIONS AND STRUCTURE OF THIS ARTICLE

This article proposes and evaluates a comprehensive set of sparsity-preserving recoding (SpaRec) strategies for sparse systematic RLNC. Section II specifies a SpaRec strategy with a decoder in each intermediate network node as well as two SpaRec strategies without decoders: one strategy with a packet buffer that can hold the recoding window size in each intermediate node and one strategy that operates with smaller packet buffers in the intermediate nodes. Also, Section II specifies a rateless (adaptive) sparse recoding approach that can be combined with any of the proposed SpaRec strategies. Section III describes the multi-hop network evaluation set-up and defines the metrics for evaluating the packet loss, in-order packet delay, as well as recoding and decoding (computation) throughput.

Section IV-A first evaluates the utilization of idle slots appearing due to packet erasures on the incoming links in the intermediate nodes. The evaluation demonstrates that sending coded packets in such idle slots can substantially reduce the packet delay. Section IV-B evaluates the impact of the code rate and the recoding window size. We find that rateless recoding with a finite recoding window size reduces the packet delay and increases the recoding and decoding throughput compared to recoding with a prescribed code rate or infinite recoding window size. Section IV-C compares the three specified SpaRec strategies operating with idle slot utilization, rateless recoding, and a finite window size against conventional recoding [67], [68], systematic RLNC [60]–[66], and a conventional recoding with small buffer benchmark [14]. We find that the proposed SpaRec strategies substantially reduce the packet losses while reducing the mean in-order packet delays down to approximately half of the benchmarks. Also, the SpaRec approaches can double the recoding throughput, and substantially increase the decoding throughput.

TABLE 1. Summary of main notations.

Symbol	Definition
S	Source node
N_n	Intermediate node n
G	Generation length (size) in # of packets
Pn	Source packet n, $n = 0, 1, 2, \ldots$
Cc	Coded packet c, $c = 0, 1, 2, \ldots$ generated by source
w	Coding window size, $w \leq G$
ϵ	Packet erasure probability
m	Number of source packets in a subset $m \leq G$
k	Number of coded packets transmitted after a subset
c	Code rate $= m/(m+k)$
δ	Delay of individual packets

II. SPECIFICATIONS OF SPARSE RECODING (SPAREC) APPROACHES

This section explains our novel recoding algorithms that maintain the sparsity of the systematic RLNC coding in multi-hop networks: Decode-Recode (Dec-Rec) in Section II-A, Recoding without Decoding (Rec-woDec) in Section II-B, and Sparsified Pure Recoding (SparsePR) in Section II-C. Section II-D specifies an adaptive recoding strategy that does not enforce a prescribed code rate and can be combined with any recoding algorithm.

A. PROPOSED DECODE-RECODE (DEC-REC) SPARSE RECODING

1) OVERVIEW

The general components of the Decode-Recode (Dec-Rec) recoding technique are: each intermediate node has a decoder to decode received coded packets instantly on-the-fly; and

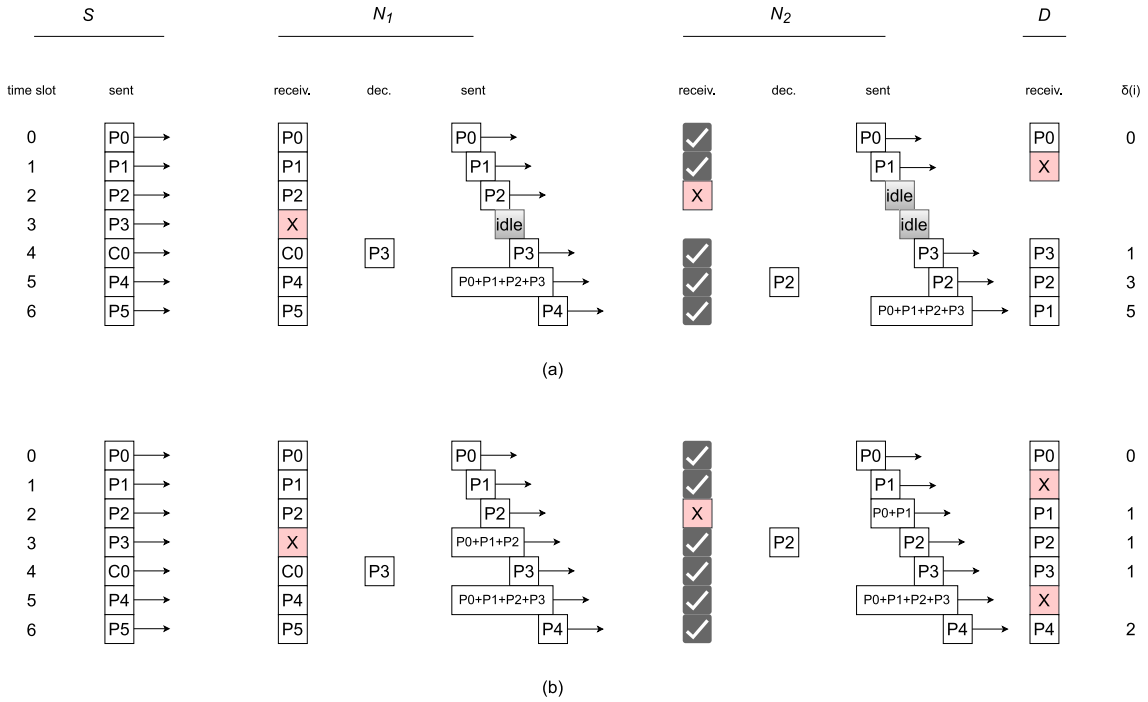


FIGURE 1. Decode-Recode (Dec-Rec) sparse systematic recoding for sliding window RLNC network coding: The code rate of the source, as well as the intermediate nodes N_1 and N_2 is $c = m/(m+k) = 4/5$. An X in the receiv. column of an intermediate node indicates a packet erasure on the link leading to the node. C is the coded packet received from the source, while P0+P1+P2+P3 denotes a recoded packet generated by an intermediate node as a combination of P0, P1, P2, and P3 with random coding coefficients. The in-order packet delay metric, defined in Section III-B2, measures the difference between the time slot when packet i is received uncoded or is decoded by the destination minus its order number i in the generation. (a) Conventional approach: *idle* time slots without new information to send go unutilized. (b) Dec-Rec with idle slot utilization: coded packets replace the *idle* slots.

each intermediate node has an encoder to generate new coded packets, i.e., recoded packets. Intermediate nodes do not need to receive an entire generation before applying recoding. With our Dec-Rec approach, recoding occurs while the packets of a generation traverse the network. Intermediate nodes may receive packets out of order due to packet erasures (or packet reordering). In order to reduce packet in-order delay, the intermediate nodes follow an algorithm similar to a code design at the source. That is, the intermediate nodes first forward a subset of the received systematic packets and interleave coded packets in between the subsets of systematic packets.

2) PACKET TRANSMISSION PATTERN

Fig 1 shows an example of the Dec-Rec scheme for two intermediate nodes N_1 and N_2 and destination D. The source node encodes the original packets using sliding window network coding [69] with a prescribed finite length encoding window [37], [65], [70]–[75], set to $w_{fin} = 5$. The first packets received by N_1, as displayed in the *receiv.* column in Fig 1, namely the Pi, $i \in 0, 1, \ldots, G-1$, are the original (systematic) source packets in a given generation, while the Cj, $j \in 0, 1, \ldots$, correspond to the coded packets generated by the source (see Table 1 for a summary of the notations). We consider that the time is slotted and in each time slot, either a systematic packet or a coded packet is transmitted over the erasure channel.

After intermediate node N_1 receives packets, it sends them in the same sliding window pattern. Specifically, N_1 first sends a subset of the earlier arrived uncoded packets or decoded packets, then N_1 generates coded packets for Forward Error Correction (FEC). In the example in Fig 1(a), the code rate of N_1 is $c = m/(m+k) = 4/5$. Therefore, after sending $m = 4$ uncoded packets (P0, P1, P2, and P3), N_1 generates a coded packet as a combination of P0, P1, P2, and P3 and sends it to the next node. Then, N_1 continues to send P4. N_2 also follows the sliding window pattern for recoding the packets.

3) PACKET DECODING AND SEEN PACKETS

Note that N_1 may be able to recover packets that were erased on the S–N_1 link by decoding received coded packets on-the-fly. An example of on-the-fly decoding is shown in the *dec.* column of N_1. After receiving coded packet C0, N_1 is able to decode P3, and then sends P3 to N_2. Another example is observed for N_2: N_2 is able to decode P2 by receiving the coded packet that is a combination of P0, P1, P2, and P3 from N_1.

If a packet erasures burst erases more than k packets, then k coded FEC packets are not sufficient to recover all erased packets. In this case, intermediate nodes have *seen* packets, which are partially decoded packets with information of future packets [76]. For sake of clarity, we give an example

of the coefficient vector of a *seen* packet. Assume that on the S–N_1 link, packet P2 would get erased, together with P3. Then, the coded packet C0 would not be able to decode P2 and P3. In the coding coefficient row $\begin{bmatrix}0 & 0 & 1 & F & 0 & 0 & 0\end{bmatrix}$, there would be a *seen* packet coding coefficient of 1 that corresponds to P2 and the F, which is a random value from the considered Galois field, corresponding to the P3. Note that since P0 and P1 are available uncoded, they would be subtracted from C0, and only the combination of P2 and P3 remains undecoded, being referred to as a seen packet [76]. When *seen* packets appear, intermediate nodes operating with Dec-Rec also send the *seen* packets to the destination. Receiving the *seen* coded packets can aid the decoding at the destination.

When a full generation is decoded by an intermediate node, then the node ignores any further packets arriving from an upstream intermediate node or the source. Then the node first checks if there are newly decoded packets that have not been previously transmitted in uncoded form and transmits all such packets in uncoded form. Subsequently, the node newly generates and transmits fully dense coded packets that are combinations of all packets in the generation until the end of the generation transmission is determined through conventional signaling, e.g., header signaling [77].

4) IDLE SLOT UTILIZATION

In a time slot with a packet erasure, there may not be new information to send. More specifically, when a packet erasure occurs, an *idle* time slot appears, as illustrated in Fig 1(a) for several examples (see X in the receiv. column followed by *idle* in the sent column). Conventionally, nodes do not send any information in such an idle time slot.

For the SpaRec algorithms, including Dec-Rec, we propose to send coded packets instead of staying *idle*. Examples of such transmissions of coded packets in conventionally idle slots are shown in Fig 1(b). We observe from Fig 1(b) that N_2 recovers P2 one time slot earlier when it receives the coded packet that is a combination of P0, P1, and P2 than with the conventional approach in Fig 1(a). Similar to N_2, the destination also receives packets earlier when *idle* time slots are filled with coded packet transmissions. For instance, P1 is recovered in time slot 6 with the conventional approach in Fig 1(a); however, P1 is recovered in time slot 2 in Fig 1(b).

5) ANALYSIS OF RECODING WINDOW SIZE

For a given packet erasure probability ϵ of the outgoing link of a given source node or intermediate node and numerator m of the code rate $c = m/(m+k)$ (which should satisfy $c \leq 1-\epsilon$ [69]), we propose to set the finite coding window size (length) of the source or finite recoding window size (length) of the intermediate node to

$$w_{\text{fin}} = m \times (1+\epsilon). \tag{1}$$

If w_{fin} yields a fractional number, we round to the closest integer. Only packets in the window are coded together.

Based on the evaluations in [65] demonstrating low packet loss probability and low computational complexity for finite coding window length at the source, we consider finite encoding window length at the source throughout this study. However, infinite and finite length coding windows have not been studied for recoding in the intermediate nodes. Therefore, we evaluate the performance characteristics of the different recoding window lengths in Section IV-B.

B. PROPOSED RECODING WITHOUT DECODING (REC-WODEC) SPARSE RECODING

1) OVERVIEW

With Recoding without Decoding (Rec-woDec), intermediate nodes do not decode incoming coded packets. Therefore, erased packets cannot be recovered at the intermediate nodes. An intermediate node first sends m uncoded packets; whereby, new coded packets, i.e., recoded packets, are created and sent if there are no received uncoded packets. Then, the intermediate node generates k coded packets from the previously received packets, irrespective of whether the received packets are systematic packets or coded packets. Since the code rate c of each node can be different, the numbers m and k of uncoded and coded packets, respectively, can be different at each node. Intermediate nodes can distinguish uncoded packets from coded packets by examining the coefficient vector of received packets: uncoded packets have only one non-zero element in the coefficient vector, while coded packets have more than one non-zero elements in their coefficient vector.

2) PACKET TRANSMISSION PATTERN

Figure 2 shows an example of the Rec-woDec technique. C0 and C1 are coded packets generated by the source, S0, S1, and S2 are coded packets generated by the N_1, while Z0, Z1, Z2, and Z3 are coded packets generated by N_2. The code rates of the source, N_1, and N_2 are $c_S = 3/4$, $c_{N_1} = 2/3$, and $c_{N_2} = 3/4$, respectively. Figure 2 looks similar to Figure 1; however, with Rec-woDec, received coded packets are also combined with received uncoded packets to generate new coded packets. In Figure 2, we can consider finite length recoding windows at N_1 and N_2, or infinite length recoding windows.

3) FINITE RECODING WINDOW LENGTH

Suppose the window lengths of the source, N_1, and N_2 are 4, 3, and 4, respectively. As shown in Figure 2, N_1 sends $k = 1$ coded packet for FEC after every $m = 2$ uncoded packets since the code rate is $c = 2/3$. After sending P0 and P1, N_1 generates the coded packet S0 in time slot 2. Since in this time slot, P0, P1, and P2 are in the buffer, S0 becomes a combination of these three packets. When P0, P1, and P2 are combined, each of them is multiplied with a different random coding coefficient, following the general principles of RLNC recoding [77], [78]. The coding coefficients and the payloads are summed separately. Then, the summed cod-

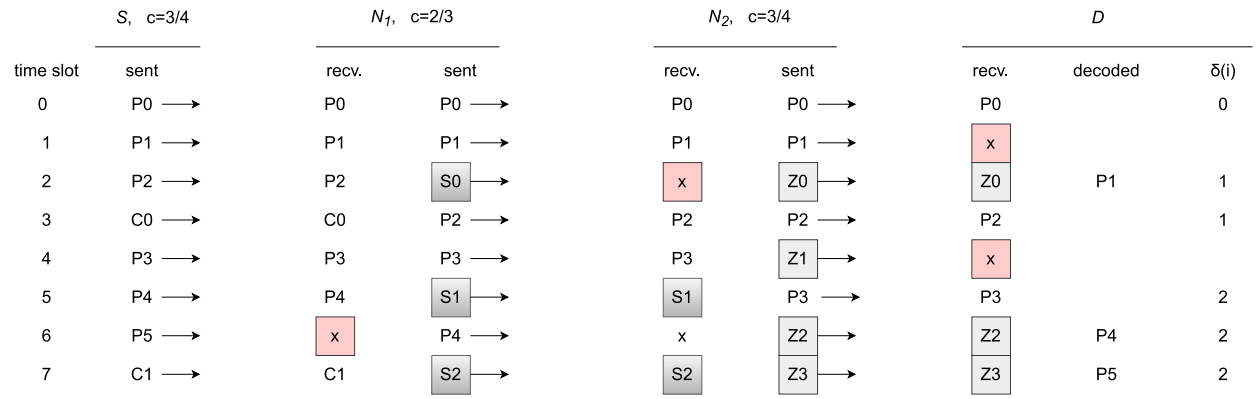


FIGURE 2. Recoding without Decoding (Rec-woDec) with idle slot utilization in the intermediate nodes: For a prescribed code rate $c = m/(m+k)$, intermediate nodes strive to follow the transmission pattern of m uncoded packets followed by k coded packets. If there is a lack of uncoded packets, then coded packets are sent. With a finite recoding window w_{fin}, the coded packets combine the last w_{fin} received packets (irrespective of whether these w_{fin} packets are uncoded or coded).

ing coefficients are appended to the payload to be sent to the next node.

Similar to S0, coded packet S1 is also a combination of the last three received packets which are P4, P3, and C0. The coded packet C0 includes P0, P1, P2; as a result, S1 becomes a combination of P0, P1, P2, P3, and P4. Although the recoding window length of N_1 is $w_{\text{fin}} = 3$, there is information of 5 packets in coded packet S1. This property is different from Dec-Rec, namely coded packets are not combined with uncoded packets in Dec-Rec. Thus, the number of non-zero elements in the coefficient vector of generated coded packets in Rec-woDec can exceed the recoding window length w_{fin}; whereas, in Dec-Rec the number of non-zero elements in the coefficient vector of the generated coded packets exactly equals the window length w_{fin}.

The coded packet S2 is generated at N_1 since there is no uncoded packet available to send; S2 is a combination of C1, P4, and P3. In turn, C1 is a combination of P2, P3, P4, and P5; as a result, S2 is a combination of P2, P3, P4, and P5. Although P5 is erased and not available uncoded at N_1, S2 still has information of P5 thanks to C1.

Examining N_2 in Figure 2, we see that only Z1 is generated for FEC, while Z0, Z2, and Z3 are generated to fill the *idle* slots. After D receives Z0, it is able to decode P1 because Z0 is a combination of P0 and P1. Although the recoding window size is $w_{\text{fin}} = 4$ in N_2, only P0 and P1 are available for recoding in time slot 2. Afterwards, P2, Z1, and P3 are sent by N_2.

Subsequently, D decodes P4 by receiving Z2 which is a combination of S1, P3, P2, and P1 (whereby S1 is a combination of P0, P1, P2, P3, and P4); hence, Z2 is a combination of P0, P1, P2, P3, and P4. Finally, D is able to decode P5 by receiving Z3, which is a combination of S2, S1, P3, and P2, whereby S2 is a combination of P2, P3, P4, and P5; hence Z3 is a combination of P0, P1, P2, P3, P4, and P5.

4) INFINITE RECODING WINDOW LENGTH

If an infinite-length recoding window is used, then the S and Z coded packets will be generated by using all received packets of a given generation, i.e., at most G packets. As an example, Z3 will be a combination of P0, P1, P2, P3, S1, and S2.

C. PROPOSED SPARSIFIED PURE RECODING (SPARSEPR)

Similar to Rec-woDec, sparsified pure recoding (SparsePR) does not have a decoder in the intermediate nodes. Also, similar to the Rec-woDec algorithm, with SparsePR, intermediate nodes send m uncoded packets followed by k coded packets as shown in Figure 2. However, the SparsePR operations in the recoder buffer are different from the Rec-woDec operations.

In Rec-woDec, every packet is stored in a different buffer as illustrated on the left-hand side of Figure 3. In order to store each packet individually, the number of packet buffers in an intermediate Rec-woDec node needs to equal the recoding window size i.e., w_{fin} for a finite recoding window size or the generation size G for an infinite recoding window size. The transmitted packets that are out of the scope of the recoding window can be discarded to open packet buffers for newly arrived and un-transmitted packets. However, the deletion of old packets causes a loss of information. SparsePR addresses this issue by keeping more information with limited packet buffers.

In SparsePR, every received packet is immediately combined with the previously received packets in the packet buffers. An example of this combination at N_1 (of Figure 2) for SparsePR with three packet buffers (3B) is shown for the first three received packets in Figure 3 on the right-hand side. When packet P0 is received, it is first sent uncoded and then stored in the recoder packet buffer Q_0. Then, packet P1 is received and sent uncoded. Afterwards, packet P1 is combined with P0 in Q_0 and then P1 is placed in the Q_1 buffer. When P2 is received, P2 is combined with the packets in the Q_0 and Q_1 buffers, then P2 is placed in the Q_2 buffer. In time slot 2, the coded packet S0 should be sent as illustrated in Figure 2. Thus, the three buffers Q_0, Q_1, and Q_2 are combined to generate coded packet S0 (for the N_1 window size of $w_{\text{fin}} = 3$). Effectively, S0 becomes the combination of P0, P1, and P2.

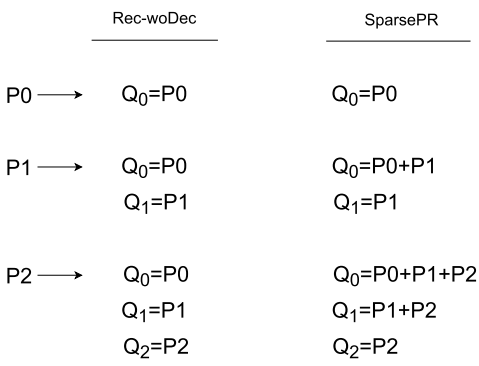


FIGURE 3. Illustration of differences of Rec-woDec and SparsePR in an intermediate node with three packet buffers (3B): Packets P0, P1, and P2 received at N_1 in Figure 2 are stored in separate packet buffers Q_0, Q_1, and Q_3 in Rec-woDec (left-hand side); whereas, SparsePR combines the packets in the packet buffers (right-hand side).

Note that an alternative way to create coded packet S0 could be to simply take the packet in Q_0 since the packet in Q_0 contains information of P0, P1, and P2. However, simply taking the packet in Q_0 for the transmission of coded packet S0 would cause linear dependencies in case multiple coded packets need to be sent, i.e., sending the packet in Q_0 more than once would not help the destination. Instead, creating different combinations from the last w packet buffers will aid the decoder in decoding erased packets.

The analogous combination steps are applied to the other received packets. Similar to S0, when coded packets S1 and S2 are generated, the last $w_{fin} = 3$ buffers are combined. When the window size is not limited, then all buffers are combined to generate coded packets; whereby the number of packet buffers is not limited for an infinite window size.

There are advantages and disadvantages of combining the received packet with the packets in the packet buffers. The advantage is that intermediate nodes can have limited numbers of packet buffers and still have the information of the previous packets. For instance, packet buffer Q_0 progressively holds information of more packets, although its capacity is for only one packet. As a result, the generated coded packets will include the information of more packets and this can reduce the delay until decoding the full generation at the decoder. One disadvantage is that this constant combination increases the computational effort in the intermediate nodes.

D. RATELESS (ADAPTIVE) SPARSE RECODING

In this section, we remove the notion of a prescribed code rate c for the SpaRec strategies in the intermediate nodes. With a prescribed code rate $c = m/(m + k)$, a node sends m uncoded packets followed by k coded packets for FEC. In a single-hop network, these k redundant coded packets are very important to recover erased packets. However, it may not be prudent to obey a prescribed code rate c in a multi-hop network. As observed in the description of Rec-woDec in Section II-B, in order to utilize the idle slots, an intermediate node can send a coded packet when there are no uncoded packets to send (even when the code rate does not yet require the transmission of a coded packet).

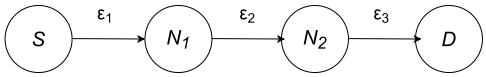


FIGURE 4. Multi-hop network model with source S, intermediate nodes N_1 and N_2, and destination D interconnected by links with packet erasure probabilities $\epsilon_1 = 0.15$, $\epsilon_2 = 0.2$, and $\epsilon_3 = 0.25$ with code rates $c_S = 8/10$, $c_{N_1} = 7/10$, and $c_{N_2} = 6/10$, unless noted otherwise.

When intermediate nodes send coded packets to comply with a prescribed code rate c and to utilize idle slots, then the destination may receive unneeded coded packets. In order to avoid this problem of excessive coded packets we propose not to obey a prescribed code rate for recoding in the intermediate nodes. Instead, we propose to forward uncoded packets whenever they arrive, and to send a coded packet when an erasure occurs (creating an idle slot) or when a coded packet arrives.

For example, consider N_1 in time slot 2 in Figure 2: after sending P0 and P1, N_1 generates and sends S0 although P2 is available in the buffer. Instead of sending S0 in time slot 2, with rateless (adaptive) recoding, N_1 sends P2 in time slot 2. The coded packet S0 is then sent in time slot 3 when C0 is received.

III. PERFORMANCE EVALUATION SETUP

This section first introduces the simulation scenario in Section III-A and the evaluation metrics in Section III-B.

A. SIMULATION SCENARIO

Figure 4 shows our experimental multi-hop network model for evaluating the packet loss and the packet delay; for the throughput evaluations, we consider one to four intermediate nodes. We model time with slots, whereby a time slot corresponds to the transmission time of one packet (as further elaborated in Section III-B2). We assume that all packets are available at the source without delay. Packet erasures are independently distributed and occur on each link with probability ϵ. We conducted 10 000 independent replications for each simulation scenario. The resulting 95% confidence intervals on the performance metrics are less than 1% of the corresponding sample means (unless noted otherwise) and are omitted from the plots to avoid visual clutter. The packet size was set to $\sigma = 1500$ bytes, which mimics the maximum size of UDP packets. Unless otherwise noted, all evaluations were conducted for a generation size of $G = 60$ packets, which is common in network coding studies [32], [79], [80]. We considered a Galois field size of $GF(2^8)$, which ensures a negligible probability of linear dependency with RLNC.

B. EVALUATION METRICS

1) PACKET LOSS

The *packet loss probability* L in percent is the difference between the generation size G and the number μ of packets that are recovered at the destination (either by being received in uncoded form or by being decoded at the destination),

normalized by the generation size G, i.e.,

$$L = \frac{(G-\mu)\times 100}{G}. \tag{2}$$

Note that all *seen* packets are counted as packet losses if they are not fully decoded by the destination.

For the loss evaluation, the source sends the number of packets that corresponds to the minimum code rate $c_{\min} = m_{\min}/(m_{\min}+k_{\min}) = 6/10$ among the code rates in the caption of Figure 4. Specifically, the source sends a total of $G + \lceil G/m_{\min}\rceil k_{\min}$ packets for a given generation. Whereby, the source follows the transmission pattern that corresponds to its source code rate $c_S = m_S/(m_S+k_S) = 8/10$ for the $\lfloor G/m_S\rfloor$ full subsets in the generation. Then, the source sends fully coded dense packets (i.e., combinations of all G source packets) until reaching a total of $G + \lceil G/m_{\min}\rceil k_{\min}$ transmitted packets. In particular, the source sends a transmission pattern of $m_S = 8$ uncoded packets followed by $k_S = 2$ coded packets $\lfloor G/m_S\rfloor = \lfloor 60/8\rfloor = 7$ times, i.e., the source sends seven full subsets, each consisting of $m_S = 8$ uncoded packets followed by $k_S = 2$ coded packets [that are combinations of $w_{fin} = 9$ [see Eqn. (1)] uncoded packets]. Then, the source sends the remaining 4 uncoded packets, followed by 26 fully dense coded packets, for a total of $\lceil G/m_{\min}\rceil k_{\min} = \lceil 60/6\rceil 4 = 40$ coded packets for the generation of $G = 60$ uncoded packets. Afterwards, the source stops. Then, the destination counts the number $G-\mu$ of received uncoded and decoded original source packets to evaluate the packet loss according to Eqn. (2). Subsequently, the source proceeds to the next generation.

2) DELAY

For a generation of G packets, the mean in-order *packet delay* D at the decoder is evaluated as:

$$D = \frac{1}{G}\sum_{i=0}^{G-1}\delta(i), \tag{3}$$

where $\delta(i)$, $i = 0, 1, 2, \ldots, G-1$, denotes the in-order delay of packet Pi in elapsed time slots. That is, $\delta(i)$ is the integer difference between the time slot in which Pi is received uncoded or decoded by the destination (whereby we neglect the computation delay for the decoding; the decoding computation delay is accounted for in the decoding throughput) and the order number i of packet Pi in the considered generation at the source.

More specifically, one integer time slot models the delivery of one packet from the source via multiple intermediate recoding nodes to the destination without erasures, as illustrated for P0 in the first line in Figs. 1(a) and (b), as well as Figure 2. We note that in real physical networks, each intermediate node incurs one time slot delay for the packet transmission in the store-compute-forward process, i.e., the actual packet delay in a real physical network corresponds to the in-order packet delay metric $\delta(i)$, plus one time slot for the transmission delay for each intermediate node, plus applicable nodal processing delays, nodal queueing delays, and link propagation delays.

Effectively, our $\delta(i)$ delay metric counts the extra time slots that are incurred due to the coding and link packet erasures. Specifically, the delay metric $\delta(i)$ counts the extra time slots due to the coding at the source, the recoding in the intermediate nodes, as well as the packet erasures on the links and the corresponding packet recovery through coded packet transmissions so as to deliver the packet in order position i of a generation to the destination (and neglects the source and in-network processing, queueing, transmission, and propagation delays). We adopted this delay metric to focus on the delay components that are directly affected by the coding and recoding mechanisms and the link packet erasures, while excluding ancillary delay components that are unrelated (constant with respect) to the coding and link packet erasure dynamics.

For the delay evaluation and the throughput evaluation, the source follows the transmission pattern from Section III-B1, and then sends additional fully coded dense packets until the destination has recovered all G original source packets in the generation.

3) THROUGHPUT

a: DECODER

For a generation of G packets, the *decoder throughput* T_d is evaluated as

$$T_d = \frac{G\sigma 8}{\tau}, \tag{4}$$

whereby σ is the packet size (in bytes), and whereby τ is the total decoding (computation) time of a generation. The decoder throughput is evaluated in Megabits per second. For the decoding computation time evaluation, all packets received at the destination for a given generation were available to the destination decoder when the decoding commenced.

b: RECODER

The recoder *throughput* T_r is evaluated analogously as the decoder throughput T_d. For evaluating the recoding throughput, τ represents the total computation time incurred for recoding the packets of a generation at an intermediate node.

Intermediate nodes have two packet processing stages. The first stage is the packet reading stage. When a packet is received, a time delay is incurred to ''read'' this packet. Whereby, reading a packet refers to the packet decoding in Dec-Rec and the examination of the coding coefficients to detect uncoded packets in Rec-woDec and SparsePR. Therefore, a different amount of reading time is incurred depending on the recoding approach. After reading a packet, the packet writing stage commences. In the writing stage, a coded packet is generated and transmitted to the next node, or an uncoded packet is transmitted to the next node. The total computation (processing) time for the packet reading and writing stages for a generation is represented as τ.

For the recoding computation time evaluation, all packets received at an intermediate node for a given generation

were available to the recoder when the recoding commenced. For scenarios with multiple intermediate nodes, the recoding throughputs of the individual intermediate nodes were averaged.

C. REFERENCE SCHEMES

1) CONVENTIONAL RECODING

In conventional recoding [67], [68], an intermediate node sends only recoded packets. For recoding, all received packets in the buffer for a given generation are combined, i.e., the recoding window size is limited to the generation size G.

2) SYSTEMATIC RLNC

With the systematic Random Linear Network Coding (RLNC) coding scheme for a prescribed code rate c [60]–[66], the source first sends the entire generation of G uncoded packets, followed by p coded packets, whereby $c = G/(G + p)$. Intermediate nodes follow the same scheme for the recoding. In particular, an intermediate node sends G uncoded packets, and then recovers any erased packets after receiving redundant packets using its decoder. Recovered packets are sent according to the priority indicated by their order number. After sending G uncoded packets, the intermediate node generates and sends p coded packets, whereby the window size for generating coded packets at an intermediate node equals the generation size G.

For an example of systematic RLNC, consider Figure 1(a) with a generation size of $G = 4$ and code rate of the source and intermediate nodes of $c = 4/5$. The source sends the coded packet C0 after sending $G = 4$ uncoded packets, the intermediate nodes N_1 and N_2 follow the same packet transmission pattern. First, N_1 sends the received P0, P1, and P2 to N_2; and N_2 sends the received packets P0 and P1 to the destination D. After receiving C0, N_1 recovers P3, which was erased on the link from S to N_1. After recovering P3, N_1 sends P3 to N_2; and N_2 receives P3 and sends P3 to D. After N_1 sends an entire generation of $G = 4$ uncoded packets, N_1 generates $p = 1$ coded packet from P0, P1, P2, and P3. By receiving this coded packet, N_2 decodes P2 and sends P2 to D. After sending the generation, N_2 also generates $p = 1$ coded packet from P0, P1, P2, and P3.

Note that in this illustrative example with generation size $G = 4$ and $p = 1$, the systematic RLNC dynamics equal the Dec-Rec dynamics in Figure 1(a). However, for larger generation sizes, e.g., $G = 60$, the dynamics differ since the systematic RLNC source sends first the $G = 60$ uncoded packets and then sends $p = G(1/c - 1)$ redundant packets.

If an intermediate node cannot recover all G packets in a generation, then the intermediate node only forwards the received uncoded packets and packets that the node was able to decode. Then, the node stays idle and does not send coded packets. A generation of recoded packets without recovering the full generation would require a specific protocol to signal the end of the packet transmissions from the preceding node; such a protocol is out of the scope of this study. Cases of an intermediate node not being able to decode the entire generation arise only for the packet loss evaluation scenario, see Section III-B1, which limits the number of transmitted coded packets for a generation.

3) CONVENTIONAL RECODING WITH SMALL BUFFER (CRSB)

Another way of recoding is proposed in [14], namely a conventional recoding with a small buffer size (CRSB). With CRSB, intermediate nodes have a small buffer, e.g., a buffer holding up to 4 packets. When packets are received, they are stored in the buffer. When recoded packets are generated, all the received packets in the buffer are combined. After generating a fixed number of recoded packets, the packets in the buffer are discarded. Then, the intermediate node includes the next packet into the buffer for recoding; whereby the next packets may be held in a different "holding buffer" before entering the recoding buffer, see [14] for details.

An example of CRSB is illustrated in Figure 2 at N_1 with a buffer size of 4. When P0 arrives, it is stored in the buffer and sent to N_2 (irrespective of whether it is an uncoded or coded packet). The recoding process starts when the second packet is received. Then, when P1 arrives, P1 is combined with P0, and a combination of P0 and P1 is sent in time slot 1. In time slot 2, P2 arrives and a combination of P0, P1, and P2 is sent. In time slot 3, C0 arrives and a combination of P0, P1, P2, and C0 is sent. Thus, consistent with the underlying conventional recoding, CRSB sends mainly coded packets; specifically, CRSB in an intermediate node recodes all packets, except for the first packet in a set of buffered packets, which is transmitted as it was received at the intermediate node (it could be received as a systematic packet or as a coded packet).

Since the buffer of size four packets is full at the end of time slot 3, the buffer is cleared (purged). In time slot 4, P3 is received and P3 is alone sent to N_2 since P3 is the only packet in the buffer after clearing. Then, P4 is received, and a recoded packet which is a combination of P3 and P4 is generated and transmitted. Then, packet P5 is sent by the source, but is erased on the link. The intermediate node sends another combination of P3 and P4 to fill the idle slot. In the last time slot in Figure 2, C1 is received at N_1. Then, another recoded packet is generated (from P3, P4, and C1) and send to N_2.

According to this CRSB approach, nodes generate at least 3 coded packets, which is equal to the buffer size minus one, before discarding the packets in the buffer. If *idle* time slots appear, the intermediate node generates additional recoded packets from the packets that are currently in the buffer to fill the gap. Therefore, the number of generated recoded packets for each set of buffered packets can be different.

In summary, by recoding only a small number of packets, the CRSB approach can reduce the computation effort of intermediate nodes and the destination compared to the conventional recoding. However, the CRSB approach destroys the systematic sparse structure created by the source.

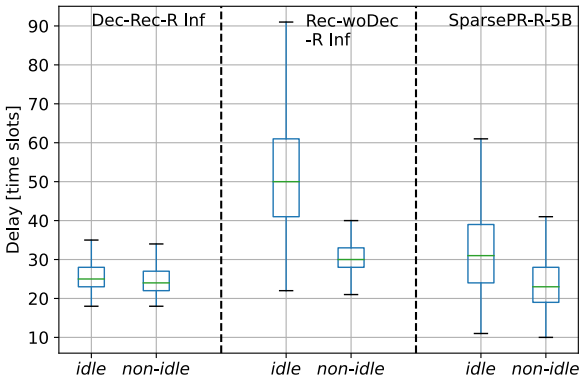


FIGURE 5. Boxplots of mean in-order packet delay D for a generation of $G = 60$ packets without utilization of *idle* slots, see Figure 1(a) (denoted by "idle"), and with utilization of *idle* slots, see Figure 1(b) (denoted by "non-idle") for three-hop network in Figure 4 with the code rates specified in the caption of Figure 4 and unlimited recoding windows.

D. IMPLEMENTATION ASPECTS

The recoding algorithms were implemented on top of the Kodo library [81] version 17.0.0. For Rec-woDec and CRSB, the packets are combined in the intermediate nodes with the Kodo fifi library. For Dec-Rec and systematic RLNC, the intermediate nodes have a decoder and an encoder utilizing the standard Kodo decoder and encoder. For conventional recoding, the regular pure-recoder of the Kodo library was used; while for SparsePR, the Kodo pure-recoder was adapted to implement the principle in Figure 2 with five packet buffers (5B, approx. half the packet buffers of the other SpaRec approaches), i.e., one packet buffer for holding the incoming packet and four packet buffers for holding packet combinations.

The throughput evaluations were performed on a computer with an Intel Core i5-6500 3.2 GHz processor and 8 Gbyte RAM operating with Ubuntu 20.04 with Linux 5.4.0.

IV. EVALUATION RESULTS

Section IV-A examines the delay implications of the idle time slot utilization. The delay and throughput of the proposed SpaRec algorithms with and without a prescribed code rate in combination with different recoding window sizes are evaluated in Section IV-B. Finally, the packet loss, in-order packet delay, and throughput performance of the best-performing SpaRec approaches are compared against the benchmarks in Section IV-C.

A. EVALUATION OF UTILIZATION OF IDLE TIME SLOTS

Figure 5 compares the mean in-order packet delay D [see Eqn. (3)] represented with boxplots for Dec-Rec, Rec-woDec, and SparsePR in the intermediate nodes without and with idle slot utilization. A boxplot shows the interquartile range from the first quartile Q_1 to the third quartile Q_3 as a box with the median marked by a horizontal line inside the box. The whiskers mark the commonly considered non-outlier range from $Q_1 - 1.5(Q_3 - Q_1)$ to $Q_3 + 1.5(Q_3 - Q_1)$.

TABLE 2. Mean throughput of a recoder at an intermediate node and the decoder at the destination for different recoding approaches with finite and infinite recoding windows; one intermediate node, packet erasure probabilities $\epsilon_1 = 0.15$ and $\epsilon_2 = 0.20$, code rates $c_S = 8/10$ and $c_{N_1} = 7/10$.

Recoding Approach	Cod. Win. Length	Throughput [Mbit/s]	
		Recoder	*Decoder*
Dec-Rec-R	Finite	385	530
	Infinite	388	516
Dec-Rec-RL	**Finite**	409	565
	Infinite	409	555
Rec-woDec-R	Finite	410	548
	Infinite	360	530
Rec-woDec-RL	**Finite**	425	583
	Infinite	373	567
SparsePR-R-5B	4 recod. pkt. bufs.	408	559
SparsePR-RL-5B	4 recod. pkt. bufs.	412	577

The coding window sizes of N_1, and N_2 were unlimited to observe the maximum impact of the *idle* case.

We observe from Figure 5 that the *idle* slot utilization (corresponding to the *non-idle* results in Figure 5) achieves substantial delay reductions for Rec-woDec and SparsePR, while not influencing the Dec-Rec delay. Dec-Rec can decode coded packets on the fly in intermediate nodes, resulting typically in a lower proportion of *idle* slots than for Rec-woDec and SparsePR, which do not decode in intermediate nodes. Also, with Dec-Rec, an intermediate node does not experience any more idle slots once its decoder can decode the full generation; then the node only sends fully dense coded packets. In contrast, using recoding without a decoder, intermediate nodes may experience idle slots up until the destination can decode the full generation.

We also note that irrespective of the *idle* slot utilization, the decoding in Dec-Rec lowers the packet delays, which is further examined in Section IV-C2. For now, we conclude that sending coded packets in the *idle* time slots decreases the delay. Therefore, we utilize *idle* slots throughout the remainder of this study.

B. IMPACT OF THE CODE RATE AND WINDOW SIZE

We observe from Figure 6 that the finite versus infinite recoding window size does not significantly affect the delay distribution. This is mainly because the finite window length w_{fin} proposed in Eqn. (1) is long enough to ensure that sufficient numbers of packets are combined in the recodings to support the timely recovery of erased packets. In particular, Eqn. (1) specifies that the finite recoding window w_{fin} covers the last

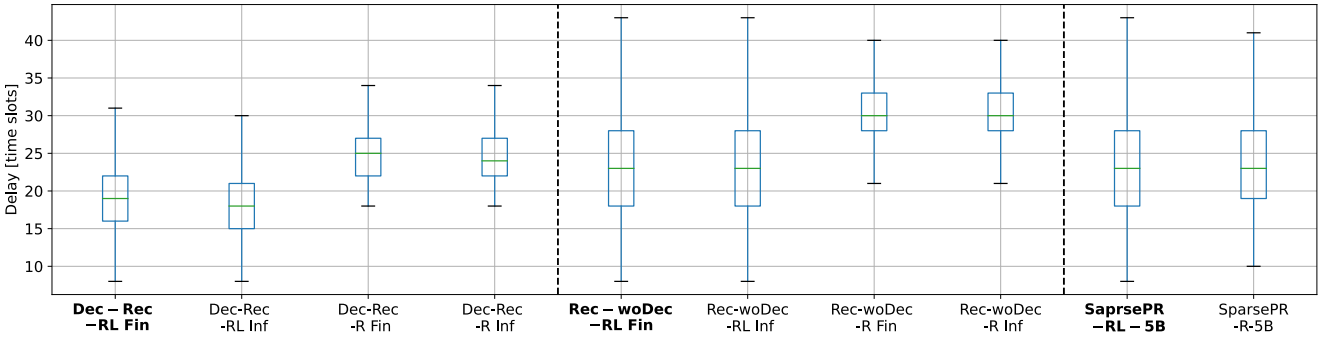


FIGURE 6. Boxplots of mean in-order packet delay D from Eqn. (3) of recoding approaches with finite and infinite recoding windows combined with a prescribed code rate or with rateless coding; for three-hop network in Figure 4.

subset of m packets, plus $m \times \epsilon$ packets (which correspond to the expected number of packets that are erased on the outgoing link of a given recoding node). Generally, the earlier the destination recovers erased packets, the lower the delay of packets that were erased on any of the links. The essentially equivalent delays for finite versus infinite recoding windows demonstrate that the finite recoding window w_{fin} supports packet loss recovery essentially as well as an infinite window length.

We also observe from Figure 6 that the rateless recoding achieves substantially shorter delays than the recoding with a prescribed code rate for the Dec-Rec and Rec-woDec approaches, while the SparsePR delays are independent of the code rate. For Dec-Rec and Rec-woDec, the shorter packet delays with rateless recoding are mainly due to the extra delays that are introduced when intermediate nodes enforce a prescribed code rate $c = m/(m + k)$ for each subset of m uncoded packets. When an intermediate node enforces a prescribed code rate c, any uncoded packets that arrive immediately after m uncoded packets have been transmitted by the intermediate node, need to wait until the intermediate node has transmitted k coded packets to fulfill the code rate c. In contrast, the rateless approach allows an intermediate node to transmit uncoded packets immediately after their arrival, lowering the packet delays. On the other hand, in SparsePR, each received packet is immediately combined with the contents of the packet buffers, see Section II-C. Then, when a coded packet is generated by combining the packet buffers, the coded packet includes information about the latest received packet. Thus, a received uncoded packet is effectively not delayed by transmitting a coded packet.

We observe from Table 2 that for all approaches, rateless recoding achieves higher recoder and decoder throughput than recoding with a prescribed code rate c. This is mainly due to an excessive generation of coded packets when intermediate nodes enforce a prescribed code rate. Specifically, coded packets are generated to fulfill the code rate c, and to fill the *idle* slots. Generating coded packets for these two purposes tends to result in superfluous coded packets that are linearly dependent to the already received packets at the destination. More specifically, the decoder evaluates the coding coefficient vector of each received packet. If a coded packet is superfluous, i.e., is not useful to recover an erased packet (because the coded packet is a linear combination of the already received packets), then the coded packet is discarded. The decoder detects a superfluous coded packet from the coding coefficient vector through a modified version of the Gauss-Jordan algorithm that detects linear dependent packets [82]. Thus, some coding vector operations are required in the recoding nodes for Dec-Rec and in the destination decoder for all SpaRec approaches to detect superfluous packets. These coding vector operations cause slight reductions of the recoding throughput and the decoding throughput.

We also observe from Table 2 that the finite recoding window length generally tends to give slightly higher throughput levels than the infinite window length. This is mainly due to the slightly higher computational complexity of decoding coded packets that are combinations of a high number of packets. With the finite window length w_{fin}, see Eqn. (1), typically on the order of ten packets are combined in a coded packet; whereas, with the infinite window, up to $G = 60$ packets are combined.

Based on the results in this section, we select the rateless (RL) recoding with finite (Fin) window length for the remaining evaluations in this article.

C. COMPARISON WITH BENCHMARKS

This section compares the performance of the SpaRec approaches Dec-Rec, Rec-woDec, and SparsePR, all operating with rateless recoding with a finite window w_{fin}, see Eqn. (1), with the benchmarks conventional recoding, systematic RLNC, and CRSB in terms of packet loss, delay, and throughput.

1) PACKET LOSS PERFORMANCE

a: PACKET LOSS DYNAMICS OF BENCHMARKS

We observe from Figure 7 that the medians and upper whiskers of the packet loss boxplots of all algorithms are zero; except for systematic RLNC. Systematic RLNC has the first

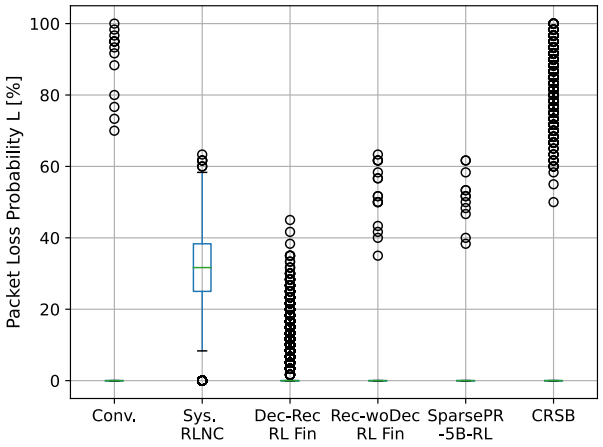


FIGURE 7. Boxplots of packet loss probability in percent of SpaRec approaches Dec-Rec, Rec-woDec, and SparsePR in comparison with benchmarks conventional recoding, systematic RLNC, and CRSB; for three-hop network in Figure 4.

packet loss quartile at around 23%, which means that 75% of the generations experienced packet losses above 23%. With systematic RLNC, the source sends the $G = 60$ uncoded packets followed by 40 coded packets. If these 40 coded packets are not enough to recover the packet erasures that occur between the source and N_1, then N_1 cannot recover the entire generation and therefore—following the systematic RLNC benchmark operation, see Section III-C2—does not send coded packets to the next node N_2. As a result, the next node N_2 would not be able to recover packet erasures that occurred between N_1 and N_2. Thus, the destination tends to experience many missing packets with systematic RLNC as observed in Figure 7.

We observe from Figure 7 that conventional recoding results in rare outliers in the 70–100% packet loss probability range (whereby most of these outliers cluster in the 90–100% range); in contrast, CRSB results in frequent outliers in the 60–100% packet loss probability range. The main reason for the outliers at these high packet loss probabilities is the relatively high number of *seen* packets (see Section II-A3) in conventional recoding. Following the recoding study in [14], CRSB applies conventional recoding with a small buffer of 10 packets [14]. In particular, with CRSB, an intermediate node transmits only coded packets (unless the buffer holds only one uncoded packet). However, the CRSB-recoded packets are combinations of fewer packets than with conventional recoding, i.e., the CRSB-recoded packets provide only a restricted (weaker) protection against packet erasures compared to conventionally recoded packets that are combinations of all packets of a generation that an intermediate node has received. Accordingly, CRSB tends to require overall more packet transmissions than conventional RLNC recoding in order to complete the decoding at the destination, as previously examined in [14, Figure 4(a)], which considers a similar recoding principle as CRSB. In our loss evaluation methodology, see Section III-B1, the source stops sending packets after a prescribed number of packet transmissions. Consequently, the event of not completing the decoding of a generation tends to occur more frequently with CRSB than with conventional recoding.

b: SPAREC PACKET LOSS DYNAMICS

The SpaRec approaches also combine only few packets, similar to CRSB. However, in contrast to CRSB, which transmits mostly coded packets, the SpaRec approaches send mostly systematic packets. Thus, the SpaRec approaches avoid the CRSB drawback of requiring a large number of coded packet transmissions for decoding. The SpaRec systematic packet transmissions also aid the decoding of seen packets on the fly; thus, mitigating the problem of conventional recoding's inability to decode the seen packets.

We observe from Figure 7 that compared to the benchmarks, the three proposed SpaRec algorithms result in outliers at lower packet loss probability levels, typically less than 60% with Rec-woDec and SparsePR, as well as typically less than 40% (with most outliers clustering below 35%) with Dec-Rec. The low packet loss probabilities of the Dec-Rec outliers are mainly due to the on-the-fly decoding at each intermediate node. Hence, Dec-Rec gives intermediate nodes a chance to recover erased packets and send them to the next node.

We observe from Figure 7 that Rec-woDec and SparsePR achieve the smallest numbers of outliers; specifically, only ten and nine of the 10 000 generations, respectively, had packets losses and these were around 40–60%. For explaining the low number of packet loss outliers of Rec-woDec and SparsePR, it is instructive to compare the dynamics of Rec-woDec and SparsePR versus the dynamics of Dec-Rec towards the end of the transmission of the packets of a generation, when the source sends fully dense coded packets (see Section III-B1). Dec-Rec independently decodes the entire generation at each intermediate node. Typically, the first intermediate node N_1 first succeeds in fully decoding the generation (usually when the source starts to send fully dense coded packets or a few time slots thereafter).

After N_1 has decoded the full generation, N_1 ignores any further packets arriving from the source. Then, N_1 first checks whether there are any newly decoded packets that have not previously been transmitted and transmits all such packets. Subsequently, N_1 generates fully dense coded packets that are combinations of all G packets in the generation. The successive intermediate node N_2 typically decodes the full generation a few time slots after N_1, and then follows the same process of transmitting packets that were not transmitted previously and then transmitting newly generated fully dense coded packets.

In contrast, when the fully dense coded packets from the source arrive to N_1, the rateless Rec-woDec and SparsePR include the fully dense coded packets in the combination of the w_{fin} packets in Rec-woDec (resp. five packets in SparsePR) that are combined to generate recoded packets. Thus, the recoded packets become fully dense coded packets. Effectively, the fully dense coded packets from the source are thus immediately forwarded (in the sense of being included

in the recoded packets) by N_1 to N_2, and similarly N_2 immediately forwards the fully dense coded packets received from N_1 to the destination. Thus, with Rec-woDec and SparsePR, the intermediate nodes begin forwarding the fully dense coded packets essentially in lock-step, i.e., in a "synchronized" fashion. This synchronized forwarding of the fully dense coded packets by the intermediate nodes to the destination, gets the fully dense coded packets to the destination sooner than with Dec-Rec, where the intermediate nodes are not synchronized. Rather, with Dec-Rec, the intermediate nodes independently (asynchronously) decode the full generation. Thus, with Dec-Rec, the fully dense coded packets tend to arrive later at the destination, resulting in a higher tendency of not finishing the decoding at the destination when the source stops transmitting.

2) PACKET DELAY PERFORMANCE

a: GENERAL RECODING PACKET DELAY DYNAMICS

Figure 8(a) shows the mean in-order packet delay of individual packets [$\delta(i)$ in Eqn. (3)], while Figure 8(b) shows boxplots of the mean in-order packet delay [D in Eqn. (3)]. Generally, we observe "ripples" in Figure 8(a) and trends of increasing delays as the packet order number i increases (up to about the middle of a generation) for all approaches, except systematic RLNC. These ripples are caused by the source packet transmission pattern in subsets, each subset consisting of $m = 8$ systematic packets followed by $k = 2$ coded packets, as per the considered source code rate $c_S = 8/10$. For the considered generation size $G = 60$, there are 7 full subsets and an additional 4 packets, resulting in 7 and a half ripples in Figure 8(a).

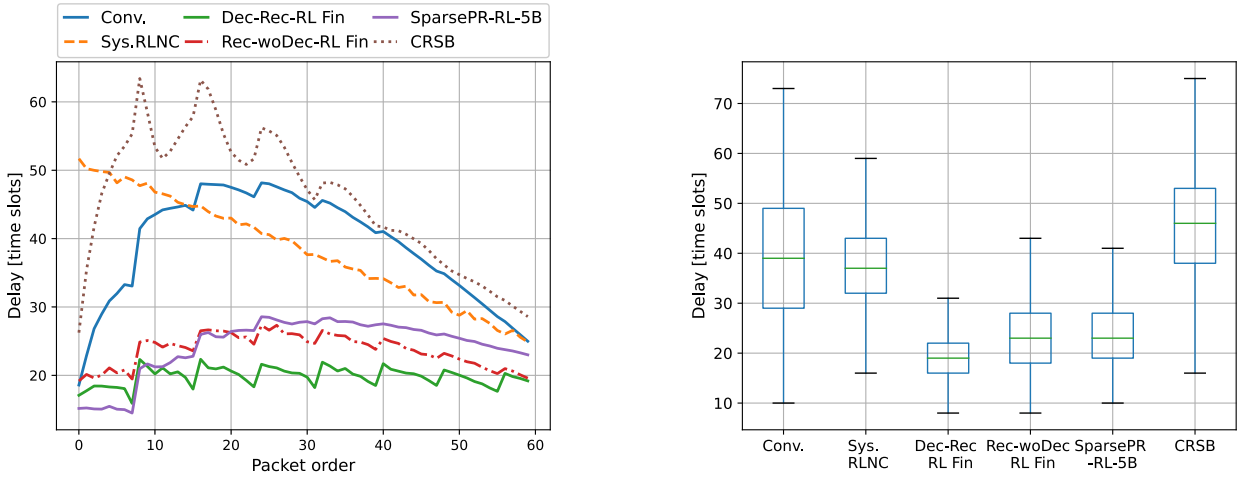
It is instructive to first consider these ripples in an erasure free network: The first $m = 8$ systematic packets Pi, $i = 0, 1, \ldots, 7$, will have a delay metric $\delta(i) = 0$. After the transmission of these first $m = 8$ packets, the source sends $k = 2$ coded packets to complete the transmission of the first subset of packets. These $k = 2$ coded packets introduce a delay for the subsequent $m = 8$ systematic packets of 2 time slots, resulting in delays $\delta(i) = 2$ for systematic packets with order numbers $i = 8, 9, \ldots, 15$ in the erasure-free scenario. The systematic packets in each subsequent subset are delayed by an additional two time slots, resulting towards the end of the generation in $\delta(i) = 12$ for $i = 48, \ldots, 55$, and $\delta(i) = 14 = \lfloor G/m_S \rfloor k_S = \lfloor 60/8 \rfloor 2$ for $i = 56, \ldots, 59$. Accordingly, the mean packet delay D across the generation of $G = 60$ packets is $D = 6.5$ in the erasure-free scenario.

It is also instructive to consider the other extreme scenario in the delay dynamics, namely the scenario when a packet Pi is erased on one of the links and only recovered at the destination when the fully dense coded packets arrive. For source code rate $c_S = m_S/(m_S + k_S)$, there are $\lfloor G/m_S \rfloor$ full subsets at the source, i.e., a total of $\lfloor G/m_S \rfloor k_S$ coded packets with finite coding window w_{fin} are transmitted by the source as part of the $\lfloor G/m_S \rfloor$ full subsets. Thus, the fully dense coded packets begin to arrive at the destination at the earliest in time slot $G + \lfloor G/m_S \rfloor k_S$. Therefore, a packet P$i$ with a delay $\delta(i) \geq G + \lfloor G/m_S \rfloor k_S - i = 74 - i$ is generally only recovered when fully dense coded packets start to arrive to the destination. (Dec-Rec is an exception since the fully dense coded packets in Dec-Rec may arrive later to the destination due to the asynchronous decoding in the individual intermediate nodes, see the end of Section IV-C1.)

We also observe from Figure 8(a) that for all approaches, except systematic RLNC, the mean in-order packet delay $\delta(i)$ is relatively short for the first packet for conventional recoding and CRSB or the first subset of $m = 8$ packets for the SpaRec approaches, and grows longer with increasing packet order number i, and becomes again shorter as the packet order number i approaches the end of a generation of G packets. This general behavior is mainly caused by the averaging of 10 000 independent replications of the transmission of the G packets ($i = 0, 1, \ldots, G - 1$) in a generation to obtain the plotted mean in-order packet delay $\delta(i)$. More specifically, packets experience typically two different delay dynamics: (a) erasure-free multi-hop transmission or "local" recovery with the coded packets for the subset that the packet belongs to, or (b) "global" recovery with the fully dense coded packets that the destination receives at the end of the generation. The erasure-free transmission or local recovery results in short delays that are on the order of the number m of packets in a subset and the total number $\lfloor G/m_S \rfloor k_S$ of coded packets with a finite coding window w_{fin}. On the other hand, the global recovery results in packet delays on the order of the "distance" $G + \lfloor G/m_S \rfloor k_S - i$ of packet Pi from the end of the generation, when fully dense coded packets arrive to the destination. For a specific example, suppose that P0 and P3 are erased on the S–N_1 link. With Dec-Rec, N_1 can recover these two erased packets when N_1 receives the $k = 2$ redundant coded packets of the first subset. With the other SpaRec approaches, the destination can similarly recover the two erased packets through on-the-fly decoding, provided no erasures occur on the other links. If one of these coded packets is erased, or a third systematic packet is erased, then these erased systematic packets require global recovery at the destination at the end of the generation.

b: PACKET DELAY DYNAMICS OF BENCHMARKS

We observe from Figure 8(a) that the in-order packet delays $\delta(i)$ of conventional recoding and CRSB increase steeply over the first $m = 8$ packets, i.e., packets $i = 0, 1, \ldots, m$, in a generation. This is mainly because conventional recoding at N_1 creates increasingly dense recoded packets as the source sends uncoded systematic packets; thus, N_1 destroys the structure of the sliding window coding scheme that the source uses. More specifically, the source first sends a subset of m uncoded packets to allow the destination to utilize these packets immediately, if they are not erased by the links. However, conventional recoding always creates coded packets which require the portion of the decoding matrix at the destination that corresponds to the packets that have been received so far to reach full rank for decoding the coded



(a) Mean in-order packet delay $\delta(i)$ as a fct. of packet order number i

(b) Boxplots of mean in-order packet delay D

FIGURE 8. In-order packet delays of SpaRec approaches Dec-Rec, Rec-woDec, and SparsePR in comparison with benchmarks conventional recoding, systematic RLNC, and CRSB for three-hop network in Figure 4: (a) mean in-order delay of individual packets $\delta(i)$ as a function of packet order number i (based on 10 000 independent replications resulting in confidence interval widths less than 4% of the means); (b) Boxplots of mean in-order packet delay D from Eqn. (3).

packets that have been received so far. Suppose that packets $0, 1, \ldots, i-1 < m$ are transmitted erasure-free and Pi suffers an erasure on the S–N_1 link or its recoded version suffers an erasure on the N_1–N_2 link or the N_2–D link; then, packet Pi cannot immediately be decoded at the destination. If a total of less than two packet erasures occur during the transmission of the first $m = 8$ packets and the subsequent $k = 2$ coded packets transmitted by the source traverse the network erasure-free, then the two packet erasures can be recovered following the RLNC principles, resulting in the downward "notch" for packet delay $\delta(7)$ (the last packet in the first subset of $m = 8$ packets) for conventional recoding in Figure 8(a).

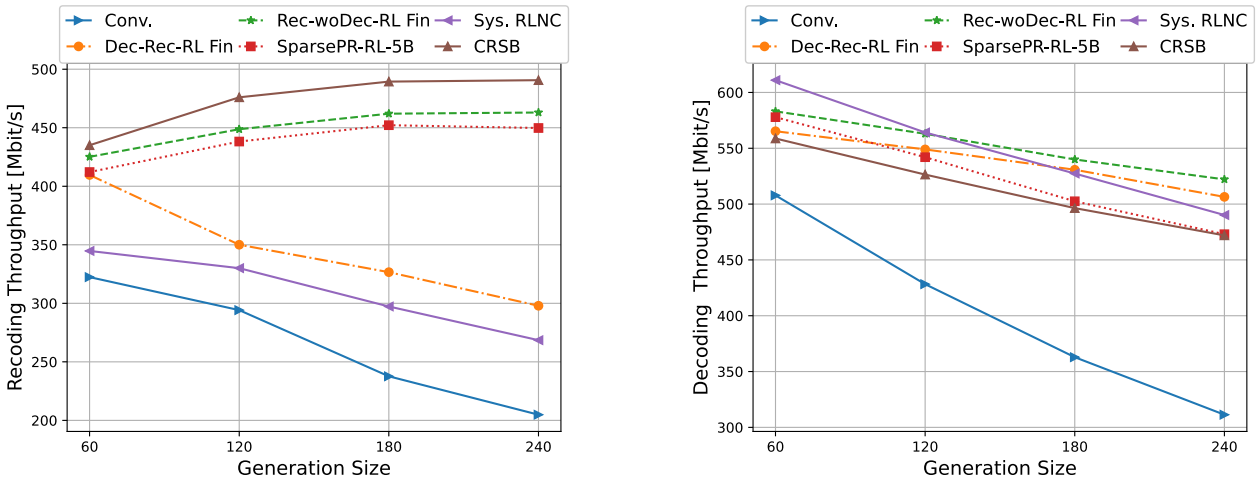
Packet erasures that cannot be recovered with the k coded packets that the source sends per subset result in *seen* packets at the destination that can only be decoded with the additional fully dense coded packets at the end of the generation. In the considered multi-hop network in Figure 4, a given packet is transmitted erasure-free over all three hops with probability $0.85 \cdot 0.8 \cdot 0.75 = 0.51$. Hence, packet erasures are quite common, resulting in the steeply growing delays $\delta(i)$ for conventional recoding in Figure 8(a), as the additional fully dense coded packets at the end of the generation are required to recover from packet erasures. Accordingly, the mean packet delays D across the G packets in a generation are very high for conventional recoding in Figure 8(b).

We observe from Figs. 8(a) and 8(b) that CRSB gives somewhat higher packet delays than conventional recoding. This in mainly due to the relatively weak protection in CRSB by combining few packets which then requires more packet transmissions at the end of a generation, confirming the results in [14, Figure 4(a)]. More specifically, in conventional recoding, all packets for a generation in the buffer in an intermediate node are combined when recoded packets are generated. CRSB only combines the packets in a small buffer to generate recoded packets. By periodically purging the buffer in the CRSB approach (see Section III-C3), intermediate nodes lose the information of previous packets. Accordingly, CRSB requires more fully dense coded packets at the end of a generation.

With systematic RLNC, any packet erasure during the transmission of the G systematic packets in the first G time slots can only be recovered with the fully dense coded packets transmitted at the end of the generation. Accordingly, the mean in-order packet delay $\delta(i)$ decreases with the "distance" to the end of the generation, as observed in Figure 8(a). Nevertheless, the systematic packet transmissions i that arrive erasure-free have an in-order packet delay $\delta(i) = 0$, reducing the upper quartile of the mean packet delay D in Figure 8(b).

c: SPAREC PACKET DELAY DYNAMICS

We observe from Figure 8(a) that the in-order packet delays $\delta(i)$ with the SpaRec schemes are substantially lower than for the benchmarks, whereby Dec-Rec achieves generally the lowest packet delays. The Rec-woDec and SparsePR packet delays are typically a few time slots higher than the Dec-Rec delays. The somewhat higher packet delays $\delta(i)$ of Rec-woDec and SparsePR compared to Dec-Rec are primarily due to the inclusion of coded packets in the packet combinations created by Rec-woDec and SparsePR, increasing the coding density of the created recoded packets. These denser recoded packets tend to cause more seen packets at the destination, which require more fully dense coded packets at the end of the generation for decoding. In contrast, Dec-Rec only creates coded packets from systematic or decoded packets. Therefore, the Dec-Rec recoded packets have a relatively lower (sparser) coding density (as examined in more detail in Table 3), causing fewer seen packets at the destination, and



(a) Recoder (b) Decoder

FIGURE 9. Mean (computation) throughput of recoder T_r and decoder T_d as a function of generation sizes G for one intermediate node and packet erasure probabilities $\epsilon_1 = 0.15$ and $\epsilon_2 = 0.2$.

thus requiring fewer fully dense coded packets at the end of the generation. Also, due to the decoding at each intermediate node, Dec-Rec tends to deliver more systematic packets to the destination than Rec-woDec and SparsePR.

Overall, we observe from Figure 8(b) that the SpaRec approaches significantly reduce the mean delay D of the packets in a generation compared to the existing recoding benchmarks. The third quartiles of the mean packet delay D with the SpaRec approaches are consistently below the first quartiles of the mean packet delays D of the existing benchmarks. The mean packet delays D of the proposed Dec-Rec approach are approximately half or less of the corresponding mean packet delays of the existing benchmarks.

In additional evaluations, which we cannot include in detail due to space constraints, we examined the sensitivity of the packet loss and delay performance to an underestimation of the erasure probability ϵ by increasing ϵ by 0.1 for each link in Fig. 4 while keeping the source code rate initially unchanged at $c_S = 8/10$ (and rateless recoding in intermediate nodes). We found that the increased ϵ: (*i*) increased the mean in-order packet delays D by a factor of 1.5 to 1.8, and (*ii*) increased the upper quartiles of the conventional recoding and CRSB packet loss probabilities to approx. 87%, while the SpaRec upper quartiles remained below 57%. Decreasing, c_S to 6/10: (*i*) reduced D to levels that are 1.2 to 1.6 above the original levels, and (*ii*) returned the packet loss probabilities to their original levels.

3) THROUGHPUT PERFORMANCE

a: IMPACT OF GENERATION SIZE G ON RECODING THROUGHPUT

Figure 9 presents the recoder (computation) throughput T_r and decoder (computation) throughput T_d as a function of the generation size G. A larger generation size G increases the recoder computations for a recoding approach that processes all packets that have been received for a given generation, namely conventional recoding. In particular, conventional recoding combines all (up to $G = 60$) packets that have been received so far for a given generation [77]. The systematic RLNC and Dec-Rec recoders strive to decode the complete generation in each intermediate node and thus incur increasing computational complexity as G increases. More specifically, the systematic RLNC and Dec-Rec recoders check in each time slot whether a new packet has been decoded.

We observe from Figure 9(a) that the CRSB, SparsePR, and Rec-woDec recoding throughput levels tends to slightly increase with increasing generation size G. These T_r increases for increasing G are mainly due to the sublinear growth of the number of superfluous packets that are generated per generation as G grows. Specifically, additional evaluations revealed for CRSB for $G = 60$ a mean of 28 superfluous packets, while the four times larger $G = 240$ had only a 2.5 times larger mean of 74 superfluous packets. A proportionally smaller number of superfluous packets implies a proportionally lower processing burden for intermediate nodes (and the destination) from superfluous packets.

b: IMPACT OF GENERATION SIZE G ON DECODING THROUGHPUT

We observe from Figure 9(b) that conventional recoding gives the lowest decoding throughput. With conventional recoding, the destination receives only dense coded packets. Dense coded packets require substantial computation effort for decoding that grows with the generation size G on the order of G^3 [79], [80] since the entire $G \times G$ coding coefficient matrix and the corresponding packet payloads need to be processed [79], [80]. Accordingly, the decoding throughput T_d [see Eqn. (4)] drops with a quadratic order, as observed in Figure 9(b). CRSB, which also delivers essentially only coded packets to the destination, achieves higher decoding throughput as only the coding coefficients corresponding to the up to 10 packets in the recoding buffer are non-zero. Thus,

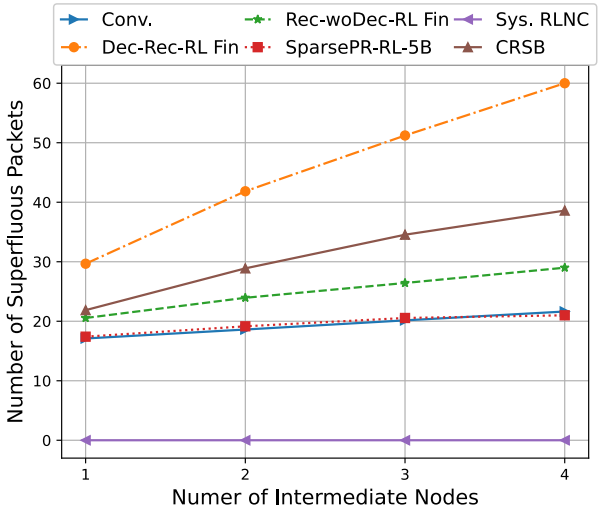


FIGURE 10. Mean number of superfluous received packets at the destination per generation of $G = 180$ packets for $\epsilon = 0.15$ for all links.

the CRSB recoded packets have a lower coding density compared to the dense coded packet of conventional recoding.

We observe from Figure 9(b) that the SpaRec approaches are clustered together and achieve nearly the same decoding throughput levels, which are in the general vicinity of the systematic RLNC decoding throughput. The SpaRec approaches and systematic RLNC achieve high decoding throughput levels mainly because of the systematic packets received at the destination.

c: IMPACT OF NUMBER OF INTERMEDIATE NODES ON SUPERFLUOUS AND SYSTEMATIC PACKETS AT DESTINATION

Before the examination of the throughput as a function of the number of intermediate nodes in Section IV-C3.d it is instructive to consider the mean number of superfluous received packets per generation at the destination as displayed in Figure 10. For a given generation, the number of superfluous received packets was evaluated by subtracting the generation size G from the total number of received packets until the generation could be decoded at the destination. Thus, effectively, Figure 10 shows the numbers of superfluous coded packets that are generated by the recoding algorithms per generation. These superfluous received coded packets are useless to the decoder at the destination in that they do not increase the rank of the decoder coefficient matrix, i.e., these superfluous received coded packets are linear combinations of previously received packets. Determining the linear dependency takes some computational effort [82].

We observe from Figure 10 that systematic RLNC has essentially no superfluous coded packets. With systematic RLNC, the source node and each intermediate node sends each of the G packets in the generation once in systematic, i.e., uncoded form, and then sends fully dense coded packets, which can be utilized to recover any erased packet of the generation at the next intermediate node or the destination.

In contrast, we observe from Figure 10 that Dec-Rec has the highest numbers of superfluous received coded packets.

TABLE 3. Mean number of uncoded packets received by the destination and coding density (proportion of non-zero coding coefficients) of the received coded packets that increase the rank of the decoder as a function of the number of intermediate nodes for $G = 180$ packets per generation and $\epsilon = 0.15$ for all links.

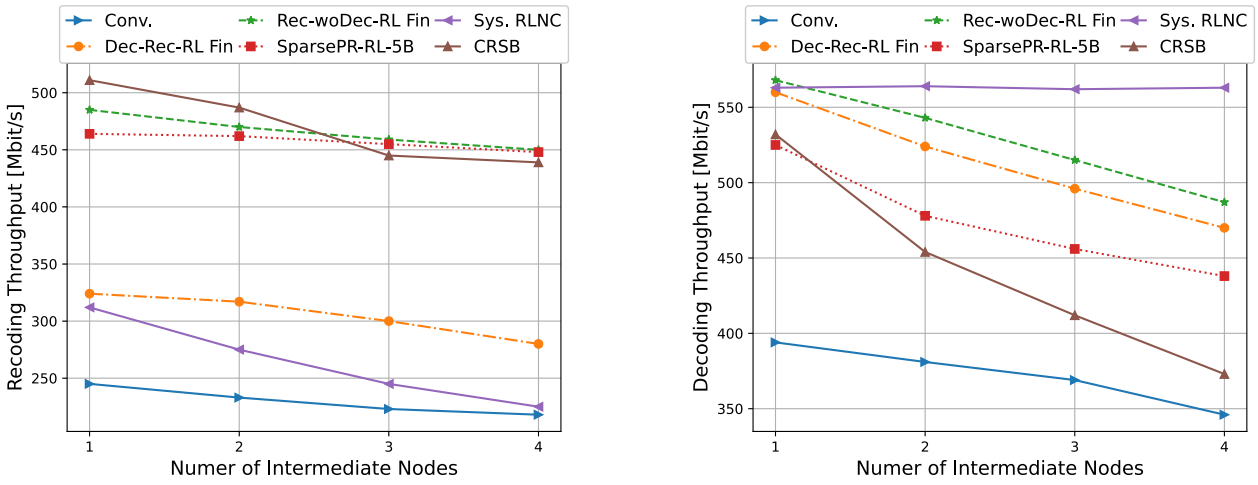
Approach	1 node	2 nodes	3 nodes	4 nodes
Conv. # of uncod. pkts.	0.867	0.799	0.706	0.632
cod. dens. of cod. pkts.	0.52	0.52	0.53	0.53
Sys. RLNC	153	153	153	153
	0.996	0.996	0.996	0.996
Dec-Rec-RL Fin	154	153	153	152
	0.19	0.20	0.21	0.21
Rec-woDec-RL Fin	130	111	94.2	80.0
	0.23	0.24	0.25	0.26
SparsePR-RL 5B	130	111	94.7	80.1
	0.56	0.56	0.56	0.57
CRSB	16	3.1	1.4	1.08
	0.12	0.18	0.23	0.27

This is mainly due to the relatively high number of linearly dependent packets that Dec-Rec generates by recoding a limited set of w_{fin} systematic and decoded packets in the intermediate nodes. Rec-woDec and SparsePR create less superfluous packets by including received coded packets when combining packets to create recoded packets. These Rec-woDec and SparsePR recoded packets therefore tend to have a higher coding density (see Table 3) and to include information from a wider set of packets, reducing the probability of linear dependency. More specifically, SparsePR has a higher coding density than Rec-woDec due to the continuous combination of each incoming packet with the packet buffers in SparsePR (see Fig. 3); therefore, SparsePR has a lower number of superfluous packets than Rec-woDec.

We observe from Table 3 that the recoding approaches that strive to build up the full generation of systematic (uncoded) packets at each intermediate node, i.e., systematic RLNC and Dec-Rec, deliver a high number of uncoded packets to the destination that is independent of the number of intermediate nodes. With systematic RLNC, the last intermediate node transmits all packets in uncoded form to the destination, followed by fully dense coded packets to recover the packet erasures on the last link. In contrast, Dec-Rec intersperses coded packets that are combinations over the finite coding window w_{fin} (and thus have a low coding density) among the uncoded packet transmissions. Rec-woDec also combines packets in the w_{fin} window, achieving low coding density; whereas, SparsePR codes with the packet buffer structure in Fig. 3, resulting in a moderately high coding density around 0.56. Both, Rec-woDec and SparsePR deliver more coded packets to the destination as the increased number of erasures with the increasing number of links create more idle slots. Similarly, with more links, the mean coding density of the delivered coded CRSB packets increases, mainly due to the increasing number of fully dense coded packets at the end of a generation.

d: IMPACT OF NUMBER OF INTERMEDIATE NODES ON THROUGHPUT

Figure 11 presents the recoder throughput T_r and decoder throughput T_d as a function of the number of intermediate



(a) Recoder (b) Decoder

FIGURE 11. Mean (computation) throughput of the recoder T_r and decoder T_d as a function of number of intermediate nodes; fixed parameters: generation size $G = 180$ packets, erasure probability $\epsilon = 0.15$ for all links.

nodes. We observe from Figure 11(a) that all approaches exhibit decreasing recoding throughput as the number of intermediate recoding nodes increases, whereby this decrease is most noticeable for CRSB and systematic RLNC.

As the number of links that the packets have to traverse increases, the probability of packet erasures on the set of links increases, as each link erases a packet independently with probability ϵ. Accordingly, Dec-Rec, which strives to decode the entire generation in each intermediate node, requires more fully coded packets at the end of the transmission of a generation to decode the generation. Thus, when averaging over a full generation, the intermediate nodes—especially the intermediate nodes towards the end of the multi-hop path—have to decode on average more fully dense coded packets, which requires more computational effort for decoding. Therefore, the plotted Dec-Rec average recoding throughput (across the given set of intermediate nodes) decreases as the number of intermediate nodes on the multi-hop path increases.

Subtly different, in systematic RLNC, each intermediate node builds up the full generation from the systematic packets and coded packets received from the immediately preceding node along the path. Thus, the number of erased packets that need to be recovered in a given intermediate node is only affected by the erasures on the one link form the immediately preceding intermediate node (or source). However, the order in which the packets arrive to the individual intermediate nodes is more strongly re-shuffled as more systematic packets are erased on the successive links and each intermediate node immediately forwards the received systematic packets (one packet per time slot, in the packet order number from the generation at the source, see Section III-C2). Packets that have been erased on the incoming link are recovered with the fully dense coded packets at the end of the generation. The transmission order according to the packet order number, which has been adopted to reduce the in-order packet delays, requires more extensive searching for the first in-order decoded-not-yet-transmitted packet as the number of intermediate nodes increases.

The CRSB recoder throughput in Figure 11(a) and decoder throughput in Figure 11(b) mainly decrease due to the relatively large number of fully dense coded packets that CRSB has to recode in the intermediate nodes at the end of a generation. CRSB offers weak protection against packet erasures, as examined in Section IV-C1.a, and, commensurately, requires a high number of fully dense coded packets at the end of the generation (approx. 11 and 28 such packets for one and four intermediate nodes, respectively). The required high number of fully dense coded packets can be inferred from the high number of superfluous received coded packets with CRSB in Figure 10. More specifically, many of the recoded packets that CRSB generates in the intermediate nodes by combining the packets in the buffer during the transmission of the finite-window coded packets from the source are linearly dependent as they do not include erased packets from outside the range of currently buffered packets (and the buffer is periodically purged, thus on average only approximately 5 packets are combined for the considered 10 packet buffer). And the probability of packet erasures from outside the buffered range increases as packets are independently erased over a larger set of links. Therefore, the recovery at the destination requires the transmission of an increasing number of fully dense coded packets at the end of a generation as the number of intermediate nodes increases, resulting in the increasing mean coding density observed in Table 3. This increasing number of fully dense coded packets poses a relatively high computational load on the intermediate nodes for recoding, leading to the declining CRSB recoding throughput over a full generation in Figure 11(a). The computational burden from the increasing number of fully dense coded packets is even higher for the decoding in the destination, leading to the

steep drop-off of the CRSB decoding throughput to near the levels of conventional recoding in Figure 11(b) as the number of intermediate nodes increases.

On the other hand, Rec-woDec, SparsePR, and conventional recoding do not periodically purge the recoding buffer. Rather, they combine all received uncoded and coded packets within the full recoding window w_{fin} (or the set of five packet buffers in SparsePR). This reduces the probability of creating linear dependent coded packets, leading to the moderate numbers of superfluous packets in Figure 10 that increase very little with the number of intermediate nodes. Thus, Rec-woDec, SparsePR, and conventional recoding avoid the large numbers of fully dense coded packets that CRSB requires at the end of a generation (e.g., Rec-woDec requires approx. 7.6 and 13 such packets for one and four intermediate nodes, respectively). Correspondingly, Rec-woDec, SparsePR, and conventional recoding do not suffer from a pronounced decrease of the recoding throughput in Figure 11(a) as the number of intermediate nodes increases.

However, with an increasing number of links, the probability that a systematic packet is transmitted erasure-free over all links to the destination decreases. Thus, the SpaRec intermediate nodes need to fill more idle slots with coded packets, which increases the computational burden, and, in turn, reduces the recoding throughput. This recoding throughput reduction is least pronounced for SparsePR, which combines each arriving packet with the packet buffers; thus, the creation of coded packets is a relatively very small additional computational load. The proportion of coded packets arriving to the destination increases as Rec-woDec and SparsePR fill more idle slots with coded packets, see Table 3. Commensurately, the decoding throughput levels for Rec-woDec and SparsePR are reduced in Figure 11(b) for an increasing number of intermediate nodes. Despite high numbers of superfluous received coded packets, see Figure 10, Dec-Rec achieves a relatively high decoding throughput, see Figure 11(b), mainly because Dec-Rec delivers a relatively high number of systematic packets to the destination decoder and the delivered coded packets have low coding density, see Table 3.

e: SUMMARY OF THROUGHPUT RESULTS

When excluding CRSB due to its poor packet loss and delay characteristics, see Figures 7 and 8, we observe from Figures 9(a) and 11(a) that Rec-woDec and SparsePR achieve the highest recoding throughput levels. Also, Rec-woDec achieves the highest decoding throughput levels [see Figures 9(b) and 11(b)]. As we observe from Figures 9 and 11, the recoding throughput is generally lower than the decoding throughput, i.e., for the same computational capabilities in intermediate nodes and the destination, the recoding in the intermediate nodes is the bottleneck. The relatively higher computational complexity for recoding compared to decoding is mainly due to the computational effort for creating recoded packets to fill the idle slots that arose due to link packet erasures, including the computations for generating the pseudo-random numbers for the recoding.

The SpaRec approaches with the highest recoding throughput levels, namely Rec-woDec and SparsePR, achieve substantially higher recoding throughput levels than the highest recoding throughput benchmark: For instance, for a generation size of $G = 240$, Rec-woDec recodes approximately 463 Mbit/s compared to about 270 Mbit/s with systematic RLNC in Figure 9(a). For four intermediate nodes in Figure 11(a), Rec-woDec recodes approximately 450 Mbit/s compared to 218 Mbit/s with conventional recoding, thus Rec-woDec can double the recoding throughput compared to conventional recoding.

V. SUMMARY AND FUTURE WORK

Within the context of sparse systematic RLNC, we have proposed and evaluated a set of three distinct sparsity-preserving recoding (SpaRec) strategies: Dec-Rec for recoding with a decoder at each intermediate network node, Rec-woDec for recoding without a decoder and with sufficient buffers to hold all packets in the recoding window, as well as SparsePR for recoding without a decoder and with limited buffers. These three SpaRec strategies can operate with a finite recoding window size or with an infinite recoding window (which then extends to all the packets in a generation). Also, the SpaRec strategies can operate with a prescribed code rate or conduct adaptive (rateless) recoding without a prescribed code rate.

Our extensive discrete-event simulation based evaluations indicate that the practical finite-length recoding window enhances the recoding and decoding (computation) throughput. Also, the rateless recoding reduces the in-order packet delays. We have compared the SpaRec strategies with finite-length recoding window and rateless recoding against several benchmarks, namely conventional recoding, systematic RLNC recoding, and conventional recoding with small buffers. The benchmark comparisons indicate that the SpaRec approaches substantially reduce the packet loss probability, reduce the in-order packet delays (to nearly half of the benchmark delays), while enhancing the recoding throughput (can be doubled) and the decoding throughput.

Among the SpaRec approaches, Rec-woDec and SparsePR achieve the lowest packet loss probabilities, nearly the lowest packet delays, and the highest recoding throughput levels in the intermediate nodes. We also find that Dec-Rec is highly competitive, achieving the lowest in-order packet delays. Interestingly, the decoding in the intermediate nodes in Dec-Rec only moderately reduces the recoding throughput while achieving nearly the same decoding throughput compared to Rec-woDec and SparsePR.

There are several important directions for future research on sparsity-preserving recoding (SpaRec) for sparse systematic RLNC. The present initial SpaRec study has focused on single-path multi-hop networks. Future research should extend the SpaRec strategies to multi-path multi-hop networks [83]–[85]. Multi-path networks pose several new challenges, such as different delays and erasure probabilities on the different paths that need to be accounted for in the coding of the packet transmissions for the different paths. Generally,

in order to avoid the complications of recoding in the intermediate nodes on the multiple paths, recoding strategies without a recoder may be a good initial starting point for researching recoding in multi-path settings.

Another important future research direction is to examine the energy consumption of the recoding algorithms. Commonly, hardware-based solutions substantially reduce the energy consumption compared to software-based solutions [86]–[88]. Therefore, it will be important to develop and evaluate efficient SpaRec hardware modules or accelerators for intermediate network nodes.

ACKNOWLEDGMENT

An earlier version of this paper was presented in part at the Proceedings of IEEE 93rd Vehicular Technology Conference (VTC2021-Spring) [1] [DOI: 10.1109/VTC2021-Spring51267.2021.9448915] and [2] [DOI: 10.1109/VTC2021-Spring51267.2021.9449001].

REFERENCES

[1] E. Tasdemir, J. A. Cabrera, F. Gabriel, D. You, and F. H. P. Fitzek, "Sliding window RLNC on multi-hop communication for low latency," in *Proc. IEEE 93rd Veh. Technol. Conf. (VTC-Spring)*, Apr. 2021, pp. 1–6, doi: 10.1109/VTC2021-Spring51267.2021.9448915.

[2] E. Tasdemir, M. Tomoskozi, H. Salah, and F. H. P. Fitzek, "Low-latency sliding-window recoding," in *Proc. IEEE 93rd Veh. Technol. Conf. (VTC-Spring)*, Apr. 2021, pp. 1–5, doi: 10.1109/VTC2021-Spring51267.2021.9449001.

[3] E. Al-Hawri, N. Correia, and A. Barradas, "DAG-Coder: Directed acyclic graph-based network coding for reliable wireless sensor networks," *IEEE Access*, vol. 8, pp. 21886–21896, 2020.

[4] E. Benamira and F. Merazka, "Maximizing throughput in RLNC-based multi-source multi-relay with guaranteed decoding," *Digit. Signal Process.*, vol. 117, Oct. 2021, Art. no. 103164.

[5] S. Laurindo, R. Moraes, C. Montez, and F. Vasques, "Combining network coding and retransmission techniques to improve the communication reliability of wireless sensor network," *Information*, vol. 12, no. 5, pp. 184.1–184.25, Apr. 2021.

[6] S. Mohammadi, P. Pahlevani, and D. E. Lucani, "Perpetual network coding for delay sensitive applications," *Wireless Pers. Commun.*, vol. 120, no. 2, pp. 923–947, Sep. 2021.

[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] F. Zhu, C. Zhang, Z. Zheng, and A. Farouk, "Practical network coding technologies and softwarization in wireless networks," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5211–5218, Apr. 2021.

[9] A. A. Abudaqa, A. Mahmoud, M. Abu-Amara, and T. R. Sheltami, "Super generation network coding for peer-to-Peer content distribution networks," *IEEE Access*, vol. 8, pp. 195240–195252, 2020.

[10] A. A. AbuDaqa, A. Mahmoud, M. Abu-Amara, and T. Sheltami, "Survey of network coding based P2P file sharing in large scale networks," *Appl. Sci.*, vol. 10, no. 7, pp. 2206.1–2206.27, Mar. 2020.

[11] D. Li, T. Song, and Y. Yang, "Content retrieval based on prediction and network coding in vehicular named data networking," *IEEE Access*, vol. 8, pp. 125576–125591, 2020.

[12] F. Stamer, J. Andre, and S. Guenther, "Network coding—State of the art," in *Proceedings of the Seminar Innov. Internet Technol. Mobile Commun. (IITM)*. Munich, Germany: Technical Univ. of Munich, 2021, pp. 63–66.

[13] S. Wunderlich, F. H. Fitzek, and M. Reisslein, "Progressive multicore RLNC decoding with online DAG scheduling," *IEEE Access*, vol. 7, pp. 161184–161200, 2019.

[14] 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./Aug. 2018.

[15] M. Karavolos, N. Nomikos, D. Vouyioukas, and P. T. Mathiopoulos, "HST-NNC: A novel hybrid satellite-terrestrial communication with NOMA and network coding systems," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 887–898, 2021.

[16] C. V. Phung, A. Engelmann, and A. Jukan, "Error correction with systematic RLNC in multi-channel THz communication systems," in *Proc. 43rd Int. Conv. Inf., Commun. Electron. Technol. (MIPRO)*, Sep. 2020, pp. 512–517.

[17] C. V. Phung, A. Engelmann, T. Kuerner, and A. Jukan, "Improving THz quality-of-transmission with systematic RLNC and auxiliary channels," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Jun. 2020, pp. 1–6.

[18] R. Prior and A. Rodrigues, "Systematic network coding for packet loss concealment in broadcast distribution," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2011, pp. 245–250.

[19] M. Yu, N. Aboutorab, and P. Sadeghi, "From instantly decodable to random linear network coded broadcast," *IEEE Trans. Commun.*, vol. 62, no. 11, pp. 3943–3955, Nov. 2014.

[20] Y. R. Julio, I. G. Garcia, and J. Marquez, "R-IoT: An architecture based on recoding RLNC for IoT wireless network with erase channel," in *Proc. Int. Conf. Inf. Technol. Syst.*, in Advances in Intelligent Systems and Computing, vol. 1137. Cham, Switzerland: Springer, 2020, pp. 579–588.

[21] R. Gao, Y. Li, J. Wang, and T. Q. S. Quek, "Dynamic sparse coded multi-hop transmissions using reinforcement learning," *IEEE Commun. Lett.*, vol. 24, no. 10, pp. 2206–2210, Oct. 2020.

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

[23] Y. Li, B. Tang, J. Wang, and Z. Bao, "On multi-hop short-packet communications: Recoding or end-to-end fountain coding?" *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 9229–9233, Aug. 2020.

[24] D. Malak, A. Schneuwly, M. Médard, and E. Yeh, "Delay-aware coding in multi-hop line networks," in *Proc. IEEE 5th World Forum Internet Things (WF-IoT)*, Apr. 2019, pp. 650–655.

[25] 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.

[26] X. Shi, M. Médard, and D. E. Lucani, "Whether and where to code in the wireless packet erasure relay channel," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 8, pp. 1379–1389, Aug. 2013.

[27] A. Singh and A. Nagaraju, "Heuristic-based opportunistic network coding at potential relays in multi-hop wireless networks," *Int. J. Comput. Appl.*, pp. 1–12, to be published.

[28] B. N. Vellambi, N. Torabkhani, and F. Fekri, "Throughput and latency in finite-buffer line networks," *IEEE Trans. Inf. Theory*, vol. 57, no. 6, pp. 3622–3643, Jun. 2011.

[29] H. H. F. Yin, K. H. Ng, A. Z. Zhong, R. W. Yeung, S. Yang, and I. Y. Y. Chan, "Intrablock interleaving for batched network coding with blockwise adaptive recoding," 2021, *arXiv:2105.07609*.

[30] S. Feizi, D. E. Lucani, and M. Médard, "Tunable sparse network coding," in *Proc. 22th Int. Zurich Seminar Commun. (IZS)*. Zürich, Switzerland: Eidgenössische Technische Hochschule Zürich, 2012, pp. 107–110.

[31] 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.

[32] Y. Li, E. Soljanin, and P. Spasojević, "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.

[33] X. Li, W. H. Mow, and F.-L. Tsang, "Rank distribution analysis for sparse random linear network coding," in *Proc. Int. Symp. Netw. Coding*, Jul. 2011, pp. 1–6.

[34] D. Silva, W. Zeng, and F. R. Kschischang, "Sparse network coding with overlapping classes," in *Proc. Workshop Netw. Coding, Theory Appl.*, Jun. 2009, pp. 74–79.

[35] 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.

[36] W. L. Chen, F. Lu, and Y. Dong, "The rank distribution of sparse random linear network coding," *IEEE Access*, vol. 7, pp. 43806–43819, 2019.

[37] 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.

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

[39] 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.
[40] V. Nguyen, E. Tasdemir, G. T. Nguyen, D. E. Lucani, F. H. P. Fitzek, and M. Reisslein, "DSEP Fulcrum: Dynamic sparsity and expansion packets for Fulcrum network coding," *IEEE Access*, vol. 8, pp. 78293–78314, 2020.
[41] 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.
[42] H. Sehat and P. Pahlevani, "An analytical model for the partial intercept probability in sparse linear network coding," *IEEE Commun. Lett.*, vol. 24, no. 4, pp. 725–728, Apr. 2020.
[43] 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.
[44] 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.
[45] A. Zarei, P. Pahlevani, and D. E. Lucani, "An analytical model for sparse network codes: Field size considerations," *IEEE Commun. Lett.*, vol. 24, no. 4, pp. 729–733, Apr. 2020.
[46] P. Garrido, D. Lucani, and R. Agüero, "Role of intermediate nodes in sparse network coding: Characterization and practical recoding," in *Proc. VDE Eur. Wireless*, May 2017, pp. 1–7.
[47] V. Nguyen, J. A. Cabrera, S. Pandi, G. T. Nguyen, and F. H. P. Fitzek, "Exploring the benefits of memory-limited Fulcrum recoding for heterogeneous nodes," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–6.
[48] S. Yang and R. W. Yeung, "Batched sparse codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 9, pp. 5322–5346, Sep. 2014.
[49] S. Yang and R. W. Yeung, *BATS Codes: Theory and Practice, Synthesis Lectures on Communication Networks*. Williston, VT, USA: Morgan & Claypool, 2017.
[50] H. H. F. Yin, R. W. Yeung, and S. Yang, "A protocol design paradigm for batched sparse codes," *Entropy*, vol. 22, no. 7, p. 790, Jul. 2020.
[51] Z. Zhou, C. Li, S. Yang, and X. Guang, "Practical inner codes for BATS codes in multi-hop wireless networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2751–2762, Mar. 2019.
[52] Z. Zhou, J. Kang, and L. Zhou, "Joint BATS code and periodic scheduling in multihop wireless networks," *IEEE Access*, vol. 8, pp. 29690–29701, 2020.
[53] X. Xu, Y. L. Guan, and Y. Zeng, "Batched network coding with adaptive recoding for multi-hop erasure channels with memory," *IEEE Trans. Commun.*, vol. 66, no. 3, pp. 1042–1052, Mar. 2018.
[54] H. H. F. Yin, B. Tang, K. H. Ng, S. Yang, X. Wang, and Q. Zhou, "A unified adaptive recoding framework for batched network coding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 1962–1966.
[55] H. H. F. Yin and K. H. Ng, "Impact of packet loss rate estimation on blockwise adaptive recoding for batched network coding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 1415–1420.
[56] A. Badr, A. Khisti, W.-T. Tan, and J. Apostolopoulos, "Perfecting protection for interactive multimedia: A survey of forward error correction for low-delay interactive applications," *IEEE Signal Process. Mag.*, vol. 34, no. 2, pp. 95–113, Mar. 2017.
[57] A. Garcia-Saavedra, M. Karzand, and D. J. Leith, "Low delay random linear coding and scheduling over multiple interfaces," *IEEE Trans. Mobile Comput.*, vol. 16, no. 11, pp. 3100–3114, Nov. 2017.
[58] M. Karzand, D. J. Leith, J. Cloud, and M. Médard, "Design of FEC for low delay in 5G," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 8, pp. 1783–1793, Aug. 2017.
[59] Y. Li, F. Zhang, J. Wang, T. Q. S. Quek, and J. Wang, "On streaming coding for low-latency packet transmissions over highly lossy links," *IEEE Commun. Lett.*, vol. 24, no. 9, pp. 1885–1889, Sep. 2020.
[60] G. Cocco, T. de Cola, and M. Berioli, "Performance analysis of queueing systems with systematic packet-level coding," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 4524–4529.
[61] F. Gabriel, S. Wunderlich, S. Pandi, F. H. 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.
[62] F. Karetsi and E. Papapetrou, "A low complexity network-coded ARQ protocol for ultra-reliable low latency communication," in *Proc. IEEE 22nd Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Jun. 2021, pp. 11–20.
[63] D. E. Lucani, M. Medard, and M. Stojanovic, "On coding for delay—Network coding for time-division duplexing," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2330–2348, Apr. 2012.
[64] S. Pandi, F. Gabriel, J. A. Cabrera, S. Wunderlich, M. Reisslein, and F. H. Fitzek, "PACE: Redundancy engineering in RLNC for low-latency communication," *IEEE Access*, vol. 5, pp. 20477–20493, 2017.
[65] S. Wunderlich, F. Gabriel, S. Pandi, F. H. Fitzek, and M. Reisslein, "Caterpillar RLNC (CRLNC): A practical finite sliding window RLNC approach," *IEEE Access*, vol. 5, pp. 20183–20197, 2017.
[66] M. Zverev, P. Garrido, R. Aguero, and J. Bilbao, "Systematic network coding with overlap for IoT scenarios," in *Proc. Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2019, pp. 1–6.
[67] A. Cohen, G. Thiran, V. B. Bracha, and M. Médard, "Adaptive causal network coding with feedback for multipath multi-hop communications," *IEEE Trans. Commun.*, vol. 69, no. 2, pp. 766–785, Feb. 2021.
[68] T. K. Dikaliotis, A. G. Dimakis, T. Ho, and M. Effros, "On the delay advantage of coding in packet erasure networks," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2868–2883, May 2014.
[69] J. Cloud and M. Médard, "Network coding over SATCOM: Lessons learned," in *Proc. Int. Conf. Wireless Satell. Syst.*, in Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 154. Cham, Switzerland: Springer, 2015, pp. 272–285.
[70] P. J. Braun, D. Malak, M. Medard, and P. Ekler, "Multi-source coded downloads," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.
[71] P. J. Braun, M. Medard, and P. Ekler, "Practical evaluation of multi-source coded downloads," *IEEE Access*, vol. 7, pp. 120304–120314, 2019.
[72] A. Cohen, D. Malak, V. B. Bracha, and M. Medard, "Adaptive causal network coding with feedback," *IEEE Trans. Commun.*, vol. 68, no. 7, pp. 4325–4341, Jul. 2020.
[73] V. Roca, F. Michel, I. Swett, and M.-J. Montpetit, "Sliding window random linear code (RLC) forward erasure correction (FEC) schemes for QUIC, internet draft, draft-roca-nwcrg-rlc-fec-scheme-for-quic-03," Internet Eng. Task Force (IETF), Fremont, CA, USA, Tech. Rep. draft-roca-nwcrg-rlc-fec-scheme-for-quic-03, 2020.
[74] E. Tasdemir, C. Lehmann, D. Nophut, F. Gabriel, and F. H. P. Fitzek, "Vehicle platooning: Sliding window RLNC for low latency and high resilience," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Jun. 2020, pp. 1–6.
[75] E. Tasdemir, C. Lehmann, and F. H. P. Fitzek, "Joint application of sliding window and full-vector RLNC for vehicular platooning," in *Proc. IEEE 11th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2021, pp. 1429–1435.
[76] J. K. Sundararajan, D. Shah, M. Medard, S. Jakubczak, M. Mitzenmacher, and J. Barros, "Network coding meets TCP: Theory and implementation," *Proc. IEEE*, vol. 99, no. 3, pp. 490–512, Mar. 2011.
[77] D. Goncalves, S. Signorello, F. M. V. Ramos, and M. Medard, "Random linear network coding on programmable switches," in *Proc. ACM/IEEE Symp. Architecture Netw. Commun. Syst. (ANCS)*, Sep. 2019, pp. 1–6.
[78] E. Tsimbalo and M. Sandell, "Reliability of relay networks under random linear network coding," *IEEE Trans. Commun.*, vol. 67, no. 8, pp. 5230–5240, Aug. 2019.
[79] H. Shin and J.-S. Park, "Optimizing random network coding for multimedia content distribution over smartphones," *Multimedia Tools Appl.*, vol. 76, pp. 19379–19395, Oct. 2017.
[80] 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.
[81] M. V. Pedersen, J. Heide, and F. H. Fitzek, "Kodo: An open and research oriented network coding library," in *Proc. Int. Conf. Res. Netw.*, in Lecture Notes in Computer Science, vol. 6827. Berlin, Germany: Springer, 2011, pp. 145–152.
[82] J. Heide, M. V. Pedersen, and F. H. Fitzek, "Decoding algorithms for random linear network codes," in *Proc. Int. Conf. Res. Netw.*, in Lecture Notes in Computer Science, vol. 6827. Berlin, Germany: Springer, 2011, pp. 129–136.
[83] C. Han, J. Yin, L. Ye, and Y. Yang, "NCAnt: A network coding-based multipath data transmission scheme for multi-UAV formation flying networks," *IEEE Commun. Lett.*, vol. 25, no. 3, pp. 1041–1044, Mar. 2021.

[84] Z. Li, M. Xu, T. Liu, and L. Yu, "A network coding-based braided multipath routing protocol for wireless sensor networks," *Wireless Commun. Mobile Comput.*, vol. 2019, Dec. 2019, Art. no. 2757601.
[85] S. Sankar, P. Srinivasan, S. Ramasubbareddy, and B. Balamurugan, "Energy-aware multipath routing protocol for Internet of Things using network coding techniques," *J. Grid Utility Comput.*, vol. 11, no. 6, pp. 838–846, 2020.
[86] J. Acevedo, R. Scheffel, S. Wunderlich, M. Hasler, S. Pandi, J. Cabrera, F. Fitzek, G. Fettweis, and M. Reisslein, "Hardware acceleration for RLNC: A case study based on the Xtensa processor with the Tensilica instruction-set extension," *Electronics*, vol. 7, no. 9, p. 180, Sep. 2018.
[87] L. Linguaglossa, S. Lange, S. Pontarelli, G. Retvari, D. Rossi, T. Zinner, R. Bifulco, M. Jarschel, and G. Bianchi, "Survey of performance acceleration techniques for network function virtualization," *Proc. IEEE*, vol. 107, no. 4, pp. 746–764, Apr. 2019.
[88] P. Shantharama, A. S. Thyagaturu, and M. Reisslein, "Hardware-accelerated platforms and infrastructures for network functions: A survey of enabling technologies and research studies," *IEEE Access*, vol. 8, pp. 132021–132085, 2020.

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.

MÁTÉ TÖMÖSKÖZI received the M.Sc. degree from the Budapest University of Technology and Economics, Hungary, in 2013, and the Ph.D. degree from TU Dresden, in 2021. He has many years of industry experience as a Senior Software Engineer. At acticom GmbH, Berlin, Germany, he worked on network header compression implementations that also influenced his later research at TU Dresden, where he specializes in advancing header compression and network coding for next generation wireless use-cases. He is currently a Postdoctoral Researcher at Technische Universität Dresden, Germany, in the framework of the Centre for Tactile Internet with Human-in-the-Loop (CeTI), a Cluster of Excellence.

JUAN A. CABRERA received the B.Sc. degree in electronics engineering from Simon Bolivar University, Venezuela, in 2013, and the M.Sc. degree in wireless communication systems from Aalborg University, Denmark, in 2015. He is currently pursuing the Ph.D. degree with the Deutsche Telekom Chair of Communication Networks, Technical University Dresden, Germany. His special research interests include network coding, fog computing, distributed storage systems, and mobile edge cloud solutions.

FRANK GABRIEL received the Dipl.-Inf. degree in computer science from Technical University Chemnitz, Germany, in 2011. He is currently pursuing the Ph.D. degree with the Deutsche Telekom Chair of Communication Networks, TU Dresden.

DONGHO YOU received the B.S. and M.S. degrees in media IT engineering and the Ph.D. degree in broadcasting and communications engineering from the Seoul National University of Science and Technology, Seoul, South Korea, in 2012, 2014, and 2018, respectively. He was a Senior Researcher at the Deutsche Telecom Chair of Communication Networks, Technical University of Dresden, from 2018 to 2021. He is currently an Assistant Professor with Hannam University, South Korea. His current research interests include 5G communications and networks for ultra-reliable and low-latency multimedia service.

FRANK H. P. FITZEK (Senior Member, IEEE) received the Diploma (Dipl.-Ing.) degree in electrical engineering from Rheinisch-Westfälische Technische Hochschule (RWTH), Aachen, Germany, in 1997, the Ph.D. (Dr.-Ing.) degree in electrical engineering from Technical University Berlin, Germany, in 2002, and the Doctor Honoris Causa degree from the Budapest University of Technology and Economy (BUTE), in 2015. He became an Adjunct Professor at the University of Ferrara, Italy, 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. His current research interests include wireless and mobile 5G communication networks, mobile phone programming, network coding, cross layer, and energy efficient protocol design and cooperative networking. 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. 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. In 2012, he received the Vodafone Innovation Prize.

MARTIN REISSLEIN (Fellow, IEEE) received the Ph.D. degree in systems engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 1998. He is currently a Professor with the School of Electrical, Computer, and Energy Engineering, Arizona State University (ASU), Tempe, AZ, USA. 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 served as an Associate Editor-in-Chief for the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, from 2007 to 2020, an Associate Editor for the IEEE/ACM TRANSACTIONS ON NETWORKING, from 2009 to 2013, and chaired the Steering Committee for the IEEE TRANSACTIONS ON MULTIMEDIA, from 2017 to 2019. He is an Associate Editor for IEEE ACCESS, IEEE TRANSACTIONS ON EDUCATION, IEEE TRANSACTIONS ON MOBILE COMPUTING, and IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT. He serves as a Co-Editor-in-Chief for *Optical Switching and Networking* and the Optical Communications Area Editor for the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS.