IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 15, 2022, accepted May 13, 2022, date of publication May 17, 2022, date of current version May 26, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3175815

FSW: Fulcrum Sliding Window Coding for Low-Latency Communication

ELIF TASDEMIR[1], VU NGUYEN[1,2], GIANG T. NGUYEN[3,4], FRANK H. P. FITZEK[1,4], (Senior Member, IEEE), AND MARTIN REISSLEIN[5], (Fellow, IEEE)

[1]Deutsche Telekom Chair, 5G Laboratory Germany, Technische Universität Dresden, 01062 Dresden, Germany
[2]Vietnam-Korea University of Information and Communication Technology, Da Nang 50000, Vietnam
[3]Chair of Haptic Communication Systems, Technische Universität Dresden, 01062 Dresden, Germany
[4]Centre for Tactile Internet with Human-in-Loop (CeTI), Technische Universität Dresden, 01062 Dresden, Germany
[5]School of Electrical, Computer, 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, Deutsche Forschungsgemeinschaft (DFG) as part of Germany's Excellence Strategy—EXC 2050/1—Project ID 390696704—Cluster of Excellence Centre for Tactile Internet with Human-in-the-Loop (CeTI) of Technische Universität Dresden; in part by the Ministry of National Education Turkey; in part by the Ministry of Education and Training of Vietnam Project 911 under Grant 911/QD-TTg; and in part by the Alcatel-Lucent Foundation under the Eco-Friendly Coding Meets Multimedia (ECOMM) Project.

ABSTRACT Fulcrum Random Linear Network Coding (RLNC) combines outer coding in a large Galois Field, e.g., $GF(2^8)$, with inner coding in $GF(2)$ to flexibly trade off the strong protection (low probability of linear dependent coding coefficients) of $GF(2^8)$ with the low computational complexity of $GF(2)$. However, the existing Fulcrum RLNC approaches are generation based, leading to large packet delays due to the joint processing of all packets in a generation in the encoder and decoder. In order to avoid these delays, we introduce Fulcrum Sliding Window (FSW) coding. We introduce two flavors of FSW: Fulcrum Non-systematic Sliding Window (FNSW), which divides a given generation into multiple partially overlapping blocks, and Fulcrum Systematic Sliding Window (FSSW), which intersperses coded packets among the uncoded (systematic) transmission of the source packets in a generation. Our extensive evaluations indicate that FSSW substantially reduces the in-order packet delay (for moderately large generation and window sizes down to less than one fourth) and more than doubles the encoding and decoding (computation) throughput compared to generation-based Fulcrum.

INDEX TERMS Fulcrum network coding, packet in-order delay, random linear network coding (RLNC), sliding window network coding, throughput.

I. INTRODUCTION

Random Linear Network Coding (RLNC) is a Forward Error Correction (FEC) mechanisms that linearly combines source packets according to random coding coefficients in a Galois Field GF to create coded packets. The coded packets carry the random coding coefficients in the packet header. A destination can decode the coded packets from the coding coefficients without any need for additional signaling or coordination between the encoder and decoder, making RLNC attractive for a wide range of communication scenarios in general wireless networks [1]–[4] and wireless sensor network [5]–[7], as well as backhaul networks [8], [9] and content distribution networks [10]–[13].

In particular, with RLNC, the decoder can decode N source packets from any set of N received coded packets as long as the coded packets (i.e., the corresponding coding coefficients) are linearly independent. A large Galois Field, e.g., $GF(2^8)$, ensures that the coding coefficients are linearly independent with a very high probability, but requires complex computations for the decoding. Therefore, $GF(2^8)$ coded packets can reasonably only be decoded on computationally powerful destination nodes. In contrast, the small Galois Field $GF(2)$ suffers from a relatively high probability of linear dependent coding coefficients, but can be decoded with elementary Exclusive OR (XOR) operations with low computational complexity. Thus, $GF(2)$ coded packets can be decoded on

The associate editor coordinating the review of this manuscript and approving it for publication was Fung Po Tso.

nodes with little computational power, e.g., Internet of Things (IoT) actuator nodes.

Conventional RLNC is restricted to operating only in a single Galois Field and thus can either cater to destination nodes with high computation power or low computation power. In contrast, Fulcrum RLNC combines a large-field outer coding, e.g., in $GF(2^8)$, with inner coding in the small field $GF(2)$, so as to flexibly cater to both high-powered and low-powered destination nodes [14], [15]. Thus, Fulcrum RLNC enables the use of RLNC as FEC for scenarios with sets of diverse destinations that differ widely in their computing capabilities, e.g., IoT systems with high-powered gateways, and low-powered actuators. However, existing Fulcrum RLNC schemes are restricted to operate only on the basis of so-called generations, which are groups of N source packets. With generation-based Fulcrum RLNC, all N source packets in a given generation are randomly combined during encoding; hence, all coded packets for a given generation need to be received before the coded packets can be decoded to obtain the source packets. The generation-based operation introduces delays that are proportional to the generation size N.

Finite sliding window RLNC seeks to reduce the delay due to the coding structure by combing the source packets in a relatively small coding window covering w, $w < N$, source packets so as to reduce the RLNC latencies [16]–[18]. Additionally, source packets can be sent in uncoded (so-called "systematic") form to bypass the decoding, and to avoid the delays associated with encoding and decoding [19]–[23]. The systematic source packet transmissions can be FEC protected by coded packets that are combinations of the source packets in a window and are interspersed among the source packet transmissions [19]–[21], [24], [25]. To the best of our knowledge, all such existing schemes for reducing the latency in RLNC packet communication operate only with a single Galois Field size and thus can only cater to either destinations with high or low computational capabilities.

A. CONTRIBUTION AND STRUCTURE

Based on the review of the related literature in Section I-B, we develop and evaluate the first sliding window approach to Fulcrum RLNC coding so as to enable RLNC FEC to diverse sets of destinations while mitigating latencies through systematic source packet transmissions and sliding window RLNC coding. The Fulcrum Sliding Window (FSW) coding developed in Section II incorporates the sliding window mechanism in the Fulcrum inner coding. In particular, Fulcrum Non-systematic Sliding Window (FNSW) conducts the inner coding for a generation of N source packets (and associated outer coded packets) through coding multiple partially overlapping blocks and sends only coded packets. In contrast, Fulcrum Systematic Sliding Window (FSSW) transmits the source packets and outer coded packets in systematic form, interspersed by sliding window inner coded packets. In summary, our main original contributions towards developing novel network coding methodologies are:

- We design the Fulcrum Sliding Window (FSW) methodology for RLNC, the first RLNC approach that permits the flexibility of utilizing different Galois fields (catering to receivers with different computing capabilities) with a low-latency sliding window.
- We design the Non-systematic FSW flavor (called FNSW) which codes all transmitted packets
- We design the Systematic FSW flavor (called FSSW) which further reduces latencies by transmitting source packets in systematic form.

Our performance evaluations in Section III compare the two proposed FSW flavors with the low-latency single Galois Field PACE RLNC approach [25] that intersperses coded packets among the systematic source packet transmissions of a given generation and with a small-generation variation of the original generation-based Fulcrum RLNC [14]. We find that FSSW attains the short in-order packet latencies of PACE while achieving higher encoding and decoding throughput levels than PACE and conventional Fulcrum and achieving nearly as high decoding probabilities as PACE. We also find that FSW coding effectively differentiates the decoding complexity between inner decoding in $GF(2)$ and outer decoding in $GF(2^8)$. Section IV summarizes the conclusions from this first FSW study and outlines future FSW research directions.

B. BACKGROUND AND RELATED WORK

1) GENERATION-BASED FULCRUM RLNC

Fulcrum RLNC [14], [15] is a convenient RLNC coding mechanism to provide the performance characteristics of multiple Galois Fields [34] to heterogeneous receiver nodes. Recent Fulcrum RLNC research has examined the reduction of the encoding and decoding computational complexity through sparse coding with a low density of non-zero coding coefficients [26] as well as the adaptivity and energy efficiency in fog computing and smart city scenarios [27]–[30]. To the best of our knowledge, as summarized in Table 1, all existing Fulcrum RLNC related research has been limited to generation-based coding, which introduces relatively long packet delays. In contrast, we introduce Fulcrum Sliding Window (FSW) coding in this study to reduce the packet delays.

TABLE 1. Summary comparison of key properties of proposed FSW RLNC versus existing RLNC approaches.

RLNC Appr. / Key Prop.	Coding Basis	# of Galois Fields
Fulcrum [14], [15], [26]–[30]	Generation	2 different GFs
Slid. Window [16]–[18], [31]–[33]	Slid. Window	Single GF
FSW (this article)	Slid. Window	2 different GFs

2) SINGLE GALOIS FIELD SLIDING WINDOW RLNC

Sliding window RLNC [16]–[18] avoids the delays of processing full generations of packets, related research has

explored the mixing of generations [35] and batched network coding [36], [37]. The low-latency characteristics of sliding window RLNC are well suited for media streaming, which has been studied in [31]–[33]. Generally, sliding window RLNC can operate in a non-systematic mode or a systematic mode. Before delving into the review of the two modes of sliding window RLNC, we briefly note that an alterative approach to reduce packet delays is to focus on systematic source packet transmissions and to enable the recovery of erased systematic source packets through interspersed coded packets, e.g., through the so-called PACE approach [25].

a: NON-SYSTEMATIC SLIDING WINDOW

Non-systematic sliding window coding partitions the original source packets into overlapping windows (also referred to as blocks) consisting of w source packets, whereby the window size w is smaller than the generation size N. The w source packets in a given block are coded together. After generating enough coded packets for a given block, the window slides forward by at most w source packets; the more overlap, the more reliable the coding.

Non-systematic sliding window coding has been studied for different types of coding, such as digital fountain codes for streaming multimedia [38], Raptor codes for efficient video broadcasting [39], and BATS codes [40]. An analytical model for non-systematic sliding window coding has been developed in [41], while the specific application context of mobile ad hoc networks has been studied in [42], [43] and cooperative communication has been explored in [44].

b: SYSTEMATIC SLIDING WINDOW

Systematic sliding window coding transmits a subset of u source packets in uncoded (systematic) form, followed by the transmission of coded packets. These coded packets are random linear combinations of the w source packets in the (coding) window. Research has explored mechanisms for advancing the systematic sliding window with feedback from the destination to the sender [21], [45] and without such feedback [46], [47]. Throughout this study, we consider sliding window coding without feedback.

To the best of our knowledge, all existing sliding window research has considered RLNC with a single fixed Galois Field, limiting the existing sliding window RLNC approaches, e.g., to either high-powered or low-powered destinations. In contrast, we introduce Fulcrum sliding window RLNC coding to support diverse sets of destinations.

II. FSW: FULCRUM CODING WITH SLIDING WINDOW

A. BACKGROUND: GENERATION-BASED FULCRUM RLNC

The general Fulcrum coding principle is that a generation of N source packet is expanded by r expansion packets that contain redundant information to a total of $N + r$ coded packets. More specifically, the original source packets $\mathcal{P} = \{P_0, P_1, \ldots, P_{N-1}\}$ are first multiplied with outer coding coefficients $c_{\ell,j}$ that are randomly selected by the encoder from $GF(2^h)$ [14]:

$$o_\ell = \sum_{j=0}^{N-1} c_{\ell,j} \cdot P_j, \quad \ell = 1, 2, \ldots, r. \tag{1}$$

These expansion packets are concatenated with the original source packets $\mathcal{P}$ to create the set of outer coded packets $\{P_0, P_1, \ldots, P_{N-1}\} \cup \{o_1, o_2, \ldots, o_r\} = \mathcal{O} = \{O_j,\ j = 0, 1, \ldots, N-1, N, N+1, \ldots, N+r-1\}$. Then, these outer coded packets are multiplied with the inner coding coefficients $\lambda_{\ell,j}$ that are randomly selected by the encoder from $GF(2)$ to create the inner coded packets:

$$I_\ell = \sum_{j=0}^{N+r-1} \lambda_{\ell,j} \cdot O_j,\ \ell = 0, 1, 2, \ldots. \tag{2}$$

The inner coded packets I_ℓ can be recoded in the intermediate nodes following the general RLNC recoding principles. Decoders can decode the received packets with inner, outer, or combined decoding, depending on their computing power.

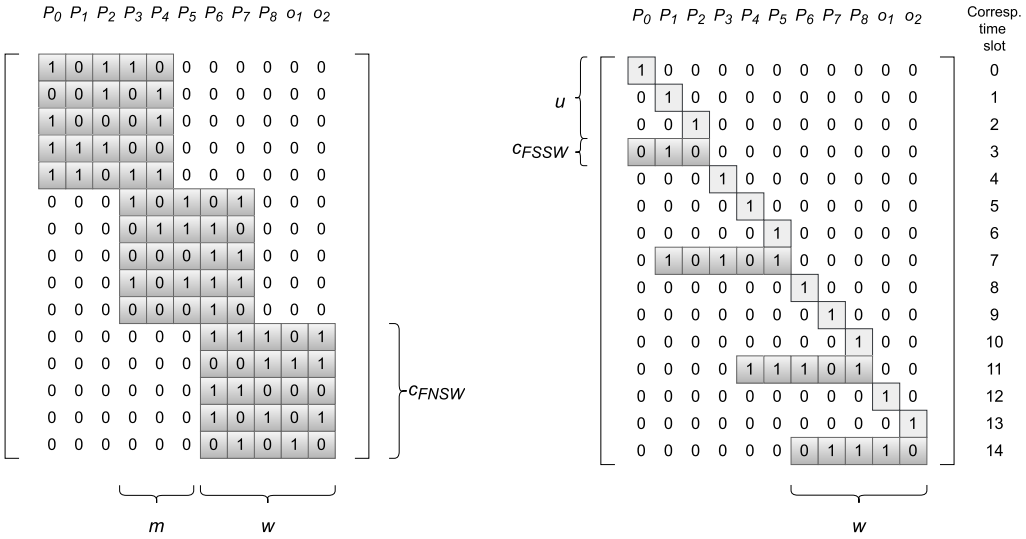
TABLE 2. Summary of main notations for Fulcrum Non-systematic Sliding Window (FNSW) and Fulcrum Systematic Sliding Window (FSSW).

Notat.	Definitions
N	Generation size in # of source packets
ϵ	Packet erasure probability on channel
r	# of Fulcrum outer expansion packets per generation
w	FNSW & FSSW: Window size
m	FNSW: Moving step in # of packets, $m \leq w$
c_{FNSW}	FNSW: # of coded packets for a window of w pkts.
t_{FNSW}	FNSW: # of slid. wind. (sparse) coded pkts. per gener.
u	FSSW: # of syst. (uncoded) packets in a subset, $u \leq N$
c_{FSSW}	FSSW: # of coded packets sent after a subset
t_{FSSW}	FSSW: # of slid. wind. (sparse) coded pkts. per gener.
$\delta(i)$	In-order delay of packet i, $i = 0, 1, \ldots, N-1$
D	Mean of packet in-order delays $\delta(i)$ for a generation

B. PRINCIPLES OF FULCRUM SLIDING WINDOW (FSW)

This section explains how non-systematic and systematic sliding window schemes can be integrated into the Fulcrum coding. The integration is applied to the Fulcrum inner coding coefficients since the number (typically at least $N + r$) of packets generated by the inner encoder is much larger than the number (r) of packets generated by the outer encoder. Also, all three types of Fulcrum decoders require the coded packets generated by the inner encoder. Hence, the structure of the inner encoding has a substantial impact on the complexity and latency of the decoding; whereas, the outer expansion packets generated by outer encoder help only the outer and combined decoders to increase the packet decoding probability, not to lower the packet latency.

In the proposed FSW coding, the outer encoder generates outer coded expansion packets in the conventional way. Specifically, r outer coded expansion packets are generated with dense outer coding coefficients from a large field size, e.g., $GF(2^8)$, see Section II-A. In FSW coding, the inner encoder randomly selects the coding coefficients in the window from $GF(2)$ and sets the other coefficients to zero.



(a) Fulcrum Non-systematic Sliding Window (FNSW)

(b) Fulcrum Systematic Sliding Window (FSSW)

FIGURE 1. Matrix of inner coding coefficients $\lambda_{\ell,j}$, whereby row index ℓ corresponds to the inner coded packet I_ℓ sent in time slot ℓ (marked on right side) and column index j corresponds to the outer coded packet O_j (original source packets P_j and $r = 2$ outer coded packets o_1 and o_2) for (a) Fulcrum Non-systematic Sliding Window (FNSW) with moving step $m = 3$, window size $w = 5$, and $c_{FNSW} = 5$ coded packets to be sent for $w = 5$ packets; and (b) Fulcrum Systematic Sliding Window (FSSW) with $u = 3$ uncoded packets per subset and $c_{FSSW} = 1$ coded packets; for generation size $N = 9$ source packets and $r = 2$ expansion packets.

C. FULCRUM NON-SYSTEMATIC SLIDING WINDOW

1) ENCODING

In order to keep the encoding and decoding computation time as well as the complexity low, Fulcrum Non-Systematic Sliding Window (FNSW) divides a data stream into *blocks*, whereby the block length equals the window length w. In the encoding, only the w packets in a given block are combined together. Figure 1(a) illustrates the coding coefficient matrix of the inner encoder for a generation of $N = 9$ source packets and $r = 2$ outer coded packets. The block length is $w \leq N$, and equivalently, the window size is $w = 5$, while the moving step is $m = 3$ packets. As illustrated in Figure 1(a), the first $c_{\text{FSSW}} \times w = 5 \times 5$ block of inner coding coefficients, which is marked with gray shading in the upper left of Figure 1(a) consists of the coding coefficients corresponding to the packets P_0, P_1, P_2, P_3, and P_4. According to the example in Figure 1(a), the encoder has to generate at least 5 coded packets out of these $w = 5$ source packets in order to enable the decoding of this block. In the example in Figure 1(a), the coding coefficient row for creating the first coded packet is 1, 0, 1, 1, 0, i.e., the first inner coded packet I_0 to be sent in time slot 0 is a combination of the source packets P_0, P_2, and P_3 (since the corresponding coding coefficients are 1), while the coding coefficients corresponding to P_1 and P_4 are zero (i.e., P_1 and P_4 are not included in I_0).

After the encoder generates c_{FNSW} coded packets from a block, the coding window is moved forward by $m = 3$ packets, as summarized in the flowchart in Figure 2. The second block consists of the inner coding coefficients corresponding to the source packets P_3, P_4, P_5, P_6, and P_7 (see grey shaded 5×5 block in the center of Figure 1(a)); whereby P_3, and P_4 are common packets in the first block and the second block. As a result, packets P_3 and P_4 are more protected than packets P_0, P_1, and P_2 since packets P_3, and P_4 are covered by the first and second blocks. In an erasure-free scenario, there is no need for overlap, i.e., the window can move by the block size $m = w$. In erasure scenarios, either a large overlap or a high number of coded packets c_{FNSW} should be set to protect against packet erasures. Note that since the inner encoder operates in $GF(2)$, the probability of creating

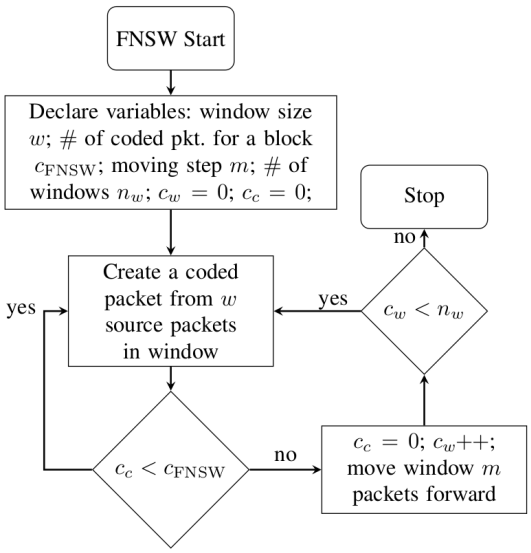


FIGURE 2. Flowchart of FNSW inner encoding: c_w counts windows, and c_c counts coded packets for a block (i.e., a window position).

linear dependent packets is high. Thus, the number of coded packets generated for one block, should at least be equal to the window size, i.e., $c_{\text{FNSW}} \geq w$.

2) DECODING

For progressive decoding, which commences elimination with the first received packet, a modified version of the Gauss-Jordan elimination is applied [48].

a: INNER DECODING

The decoding process of one block can be finalized with inner decoding when the decoder has received w linearly independent coded packets. Then, all w packets can be decoded together and sent to the upper layer. If the number of received linearly independent coded packets for one block is less than w, then the inner decoding process of this block is not successful, and additional coded packets are required. These additional coded packets can be coded packets generated by the inner encoder as random linear combinations of all $N + r$ packets. Since these additional coded packets are denser than the coded packets that are generated from a single block, they can (provided they are linearly independent) recover packet erasures or linear dependencies that occur in any block.

b: OUTER AND COMBINED DECODING

For outer or combined decoding, w linearly independent coded packets are required to decode blocks that do not include outer expansion packets (i.e., with the coefficients corresponding to the outer expansion packets set to zero). For instance, in Figure 1(a), five linearly independent coded packets are required to decode the first two blocks (since the coding coefficients corresponding to the outer expansion packets o_1 and o_2 are zero). However, the outer decoder only needs $w-r = 3$ linearly independent coded packets to decode the last block in Figure 1(a) since the dimension of the decoding matrix is $N \times N$ and the outer expansion packets are used to map back the coding coefficients to the higher field detail [14]. Moreover, these outer expansion packets are random linear combinations of N source packets, i.e., they are fully dense.

3) PARAMETER SETTINGS

The number c_{FNSW} of inner coded packets per block and the moving step m control the trade-off between low delay and low computation complexity on one hand; and protection against channel erasures on the other hand. For instance, a larger number c_{FNSW} of coded packets for a block, will increase the chance of decoding the block; on the downside, a larger c_{FNSW} will delay the transmission of the next block. We propose to set the number c_{FNSW} of coded packets for a block as a function of the channel packet erasure probability ϵ:

$$c_{\text{FNSW}} \geq w \cdot (1 + \epsilon). \tag{3}$$

A smaller moving step m implies more overlap, i.e., the window will slide forward more slowly, resulting in more blocks. Hence, a smaller m will increase the total number of generated packets and the computational time. We propose to set:

$$m \leq w \cdot (1 - \epsilon), \tag{4}$$

i.e., a larger erasure probability ϵ implies a smaller moving step m, and thus more overlap.

The number n_w of windows in one generation [40, Eq. (1)] is

$$n_w = \left\lceil \frac{N + r - w}{m} \right\rceil + 1. \tag{5}$$

We define the number t_{FNSW} of packets that are inner $GF(2)$ coded according to the sliding window scheme, i.e., as combinations of w packets (sparse coded packets), for a given generation as

$$t_{\text{FNSW}} = n_w \cdot c_{\text{FNSW}}. \tag{6}$$

For instance, in Figure 1(a), the source transmits $t_{\text{FNSW}} = 15$ packets that are coded according to the FNSW scheme. After sending these $t_{\text{FNSW}} = 15$ sparse coded packets, the source can send dense coded packets that are random linear combination of $N + r$ packets, until the N source packets can be decoded.

Based on [49, Eq. (7)], the probability of receiving w linearly independent coded packets out of c_{FNSW} transmitted inner $GF(2)$ coded packets is:

$$P(c_{\text{FNSW}}, w) = \prod_{i=0}^{w-1} \left(1 - \frac{1}{2^{c_{\text{FNSW}}-i}}\right). \tag{7}$$

D. FULCRUM SYSTEMATIC SLIDING WINDOW (FSSW)

1) ENCODING

Fulcrum Systematic Sliding Window (FSSW) first transmits a subset of u uncoded source packets, followed by c_{FSSW} coded packets, as summarized in the flowchart in Figure 3. Figure 1(b) illustrates the FSSW inner coding coefficient matrix for a generation of $N = 9$ source packets, $r = 2$ outer coded packets, subset size $u = 3$, and window length $w = 5$. As marked in Figure 1(b) with gray shading, the coding coefficient that corresponds to a systematic packet is set to one (while all other coding coefficients in the row are set to 0). For instance, in the first row of Figure 1(b), the coefficient that corresponds to packet P_0 is one and all other coding coefficients are zero, which means that P_0 is sent uncoded, i.e., systematically. After sending u systematic packets, $c_{\text{FSSW}} = 1$ coded packet is transmitted. The coding coefficients of the coded packets are selected randomly from $GF(2)$. Since $w > u$, the window also includes $w - u$ source packets from the previous subset of uncoded packets. For instance, the coverage of the second coded packet (which corresponds to time slot 7 in Figure 1(b)) includes P_1 and P_2 which were sent in the first subset. We propose to set the window length to

$$w = \lceil u \cdot (1 + \epsilon) \rceil. \tag{8}$$

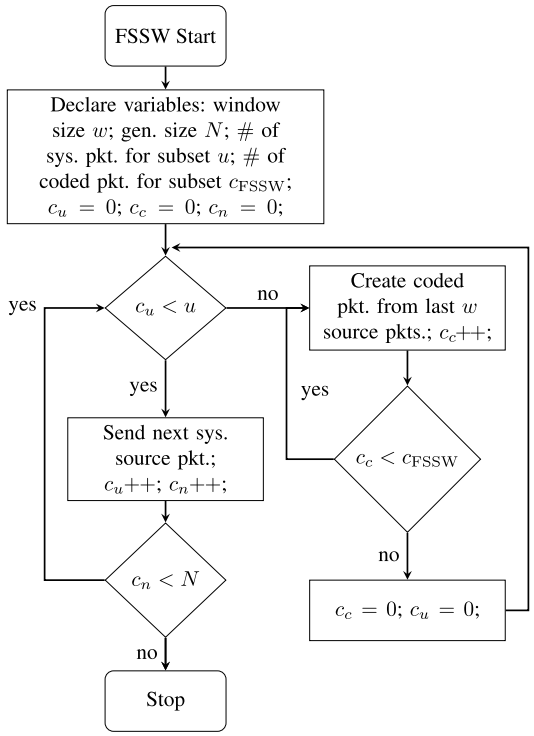


FIGURE 3. Flowchart of FSSW inner encoding: c_u counts systematic packets in a subset, c_c counts coded packets for a subset, and c_n count total transmitted systematic packets.

2) DECODING

The decoder only needs to decode coded packets to recover packet erasures since received systematic packets can immediately be sent to the upper layer. The decoder can store systematic packets to utilize in the decoding process of coded packets so as to recover packet erasures. For instance, suppose that P_5 in Figure 1(b) is erased. If the decoder receives the second coded packet which is a combination of P_1, P_3, and P_5, the decoder needs P_1 and P_3 in uncoded form to recover P_5. If a packet is erased and cannot be recovered through redundant packets, then either the sender needs to send coded packets at the end of the generation, or this packet will be counted as a loss.

3) PARAMETER SETTING

For a given packet erasure probability ϵ, the corresponding code rate $c \leq 1 - \epsilon$ [50] defines the number c_{FSSW} of coded packets that need to be transmitted for a subset of u uncoded packets:

$$c = \frac{u}{u + c_{\text{FSSW}}} \leq 1 - \epsilon. \tag{9}$$

We define the number t_{FSSW} of packets that are coded according to the FSSW sliding window scheme for a given generation as

$$t_{\text{FSSW}} = \left\lceil \frac{N + r}{c} \right\rceil = N + r + \left\lceil \frac{N + r}{u} \cdot c_{\text{FSSW}} \right\rceil; \tag{10}$$

whereby, $N + r$ packets (i.e., the N source packets and the r outer coded packets) are sent systematically by the inner encoder; and $\lceil c_{\text{FSSW}} \cdot (N + r)/u \rceil$ packets are sent as inner $GF(2)$ coded packets, i.e., as combinations of w packets (sparse coded packets).

III. PERFORMANCE EVALUATION

This section first introduces our simulation setup in Section III-A, examines the delay of individual packets in Section III-B, the encoding and decoding throughput of different generations sizes in Section III-C, the impact of different window lengths on the delay, linear dependency, and throughput in Section III-D, and the decoding probability in Section III-E.

A. EVALUATION SETUP

1) OVERALL SETUP

We implemented FSSW and FNSW with the Kodo library (kodo-fulcrum version 7.0) [51]. We measured the encoding and decoding throughput with the standard benchmarks in the library. We considered a coding scenario consisting of one sender and one receiver, without intermediate recoding nodes. A progressive decoder is applied, which is an on-the-fly version of the Gauss Jordan elimination [48], and starts decoding from the first received coded packet.

We performed the measurements in a virtual machine with two vCPUs and roughly 3 GB RAM. The properties of the host PC are Intel(R) Core(TM) i7-7600U CPU 2.90 GHz with 20 GB RAM. The size of the data packets was 1500 bytes which is the maximum size of Ethernet packets. We use the Galois Fields $GF(2)$ for the inner coding and $GF(2^8)$ for the outer coding. We conducted over 2000 independent replications for each scenario resulting in 95% confidence intervals that are less than 1% of the corresponding sample means. The confidence intervals are omitted from the plots to avoid visual clutter. The channel erases a transmitted packet independently with probability ϵ.

2) THROUGHPUT METRICS

With N denoting the generation size in number of source packets and σ denoting the packet size in bytes, the encoding throughput is defined as the generation size in bytes $N \cdot \sigma$, divided by the encoding computation time for a given generation. Similarly, we define the decoding throughput as $N \cdot \sigma$ divided by the decoding computation time for recovering the N original data packets.

3) DELAY METRICS

We assume that time is slotted between the sender and receiver, and that one packet is transmitted in each time slot. We define the packet in-order-delay $\delta(i)$ for packet i, $i = 0, 1, \ldots, N - 1$, as the index of the time slot in which the packet is decoded (while neglecting the decoding computation time, which is evaluated as decoding throughput) minus the packet order number i. For instance, suppose in

TABLE 3. FSSW and FNSW parameters for generation size $N = 64$ packets and $r = 4$ outer coded packets; window size $w = 20$ throughout.

Eras. Prob.	FSSW			FNSW		
ϵ	u	c_{FSSW}	t_{FSSW}	m	c_{FNSW}	t_{FNSW}
0.05	19	1	72	19	21	84
0.15	17	3	80	17	23	92

Figure 1(b) that P_1 is erased when it is sent uncoded in time slot 1. Then, suppose the receiver received the first coded packet in time slot 3 and recovers P_1. The in-order delay of P_1 is then two time slots, since it was supposed to be received in time slot 1, while it was actually decoded in time slot 3.

We define the mean in-order packet delay D as the mean of the in-order packet delays $\delta(i)$ experienced by the source packets i, $i = 0, 1, \ldots, N-1$, of a generation, i.e.,

$$D = \frac{1}{N} \sum_{i=0}^{N-1} \delta(i). \tag{11}$$

4) BENCHMARKS

We compare the performance of the proposed FNSW and FSSW for a given generation size N and number r of Fulcrum outer coded packets against the original generation-based Fulcrum [14] operating with a range of "small-generation" sizes η, $\eta \leq N$, and corresponding numbers ρ, $\rho \leq r$, of outer coded packets, as manipulating the generation size is a possible strategy to tune the RLNC performance [52]–[54]. Specifically, for FSW scenarios with fixed generation size $N = 64$ and $r = 4$ outer coded packets in Section III-D, we consider the following "small-generation" original generation-based Fulcrum benchmarks: transmit two small-generations, each with generation size $\eta = 32$ and $\rho = 2$ outer coded packets; four small-generations with $\eta = 16$ and $\rho = 1$; eight small-generations with $\eta = 8$ and $\rho = 1$ (so as to maintain the Fulcrum principle), and 16 small-generations with $\eta = 4$ and $\rho = 1$.

We further compare FSSW with u systematic packets in a subset and c_{FSSW} coded packets after a subset with PACE-Burst encoding [25] with u systematic packets followed by c_{FSSW} coded packets per sub-generation. The PACE benchmark focuses on low latency by interspersing $GF(2^8)$ coded packets that combine all source packets in a generation that have been transmitted so far.

B. IMPACT OF ERASURES ON PACKET DELAY

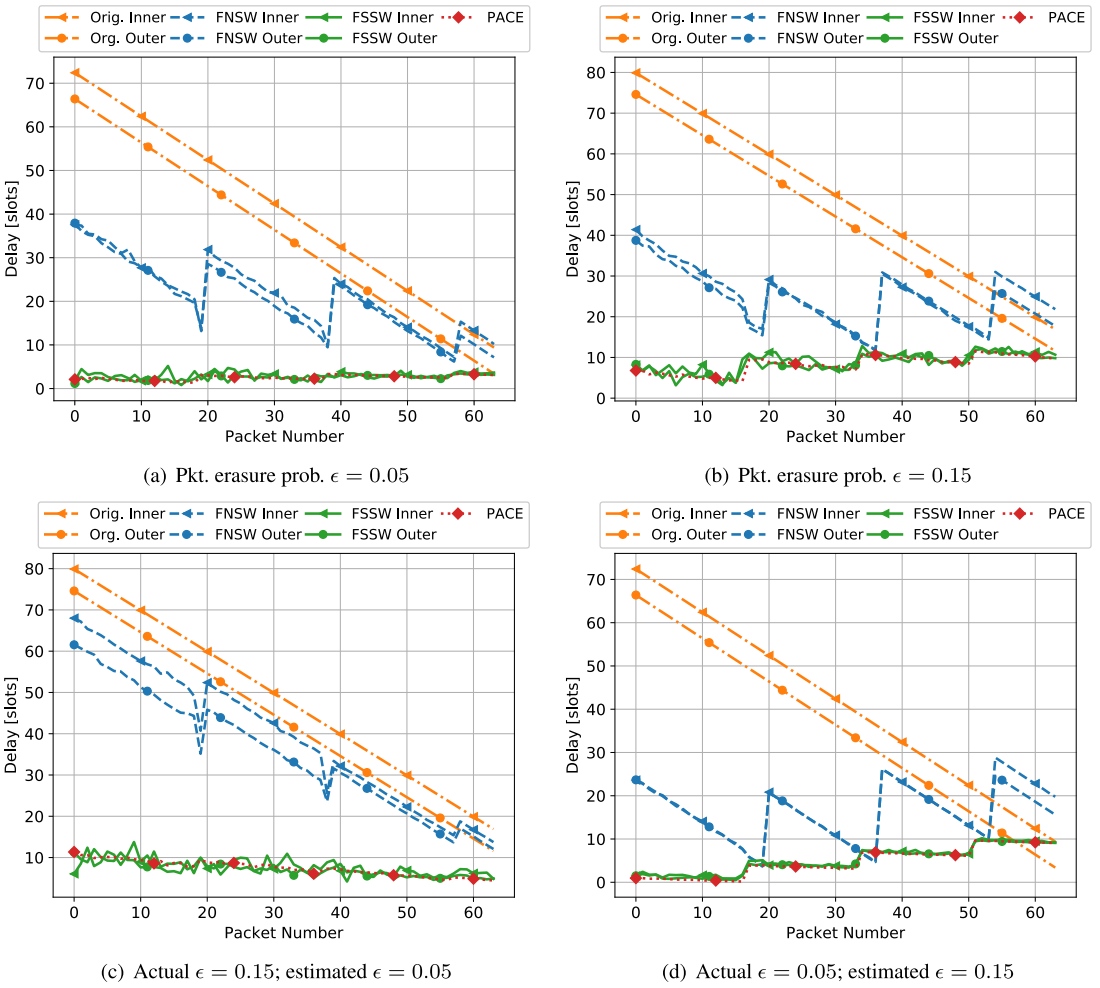
1) FSSW AND FNSW PARAMETER SETTINGS

Table 3 lists the FSSW and FNSW parameters for the two considered packet erasure probabilities $\epsilon = 0.05$ and $\epsilon = 0.15$. For both FSSW and FNSW, the code rate c was set according to Eqn. (9) for $\epsilon = 0.05$ to $c = 95/100 = 19/20$; analogously, for $\epsilon = 0.15$, we set $c = 85/100 = 17/20$. For FSSW, based on $c = 19/20$ and Eqn. (9), we set the size u of the subset of uncoded packets to $u = 19$ and the number c_{FSSW} of coded packets per subset to $c_{\text{FSSW}} = 1$; analogously, for $c = 17/20$, we set $u = 17$ and $c_{\text{FSSW}} = 3$. The window size w was set according to the FSSW approach, see Eqn. (8), and the same w value was used for FNSW. For FNSW, we set the moving step for the $\epsilon = 0.05$, $c = 19/20$ scenario to $m = 19$ and set the number of coded packets to $c_{\text{FNSW}} = 21$. For both FSSW and FNSW, after sending t_{FSSW} and t_{FSSW} packets, respectively, according to the sliding window approaches, the source sent dense $GF(2)$ coded packets until the N source packets could be decoded. These FSSW and FNSW parameters were identical for all Fulcrum decoder types, i.e., for the inner, outer, and combined Fulcrum decoders.

2) RESULTS AND DISCUSSION

Figure 4 shows the packet in-order delay $\delta(i)$ for the inner decoder as well as for the outer/combined decoder. We observe that the delays for the inner and outer/combined decoders exhibit generally similar behaviors, whereby the outer/combined decoder can slightly reduce the delays compared to the inner decoder. In particular, with original Fulcrum RLNC, the delay of the first packets (small i) is very high since all packets are decoded at the end of the generation, namely after the outer/combined decoder has received $N = 64$ linearly independent coded packets, or the inner decoder has received $N + r = 68$ linearly independent coded packets [14]. The decoded packets are sent to the upper layer. For inner decoding, the in-order delay $\delta(0)$ of the first packet is around 72 time slots for $\epsilon = 0.05$ in Figure 4(a). The decoder receives an average number of 1.6 linear dependent coded packets when coding $N + r = 68$ packets in $GF(2)$ [55, Eqn. (4)]. Accordingly, the original Fulcrum encoder has to transmit on average $(N + r + 1.6)/(1 - \epsilon) = 69.6/0.95 = 73.3$ coded packets in order to enable the inner decoding (which requires $N + r$ linearly independent coded packets) at the receiver, hence the first ($i = 0$) packet has to wait on average for the transmission of $73.3 - 1 = 72.3$ subsequent coded packets to enable the inner decoding, i.e., $\delta(0) = 72.3$; and, for the last packet, $\delta(63) = 73.3 - 64 = 9.3$. Importantly, the average (across the packets $i = 0, 1, \ldots, N-1$) of the in-order packet delays of the original generation-based Fulcrum is proportional to the generation size N.

We observe from Figure 4 that the FNSW in-order packet delay resembles a sawtooth wave with four peaks. The four peaks are due to $n_w = 4$ windows in a generation, whereby each peak corresponds to the beginning of a block. The delay of the first packet in a block is the highest among the packets in the block because in FNSW, the receiver has to have w linearly independent packets to decode the block. Thus, the first packet in a block has to wait until w linearly independent coded packets have been received, causing the decreasing slope of a given sawtooth; whereby the first (left-most) sawtooth consists of w packets and subsequent sawtooth waves consist of $m \leq w$ packets (due to the $w - m$ packets of overlap with the preceding window). If the c_{FNSW} coded packets per window of w packets are not enough to recover the erased packets of a block, then some packets in the block have to wait



(a) Pkt. erasure prob. $\epsilon = 0.05$

(b) Pkt. erasure prob. $\epsilon = 0.15$

(c) Actual $\epsilon = 0.15$; estimated $\epsilon = 0.05$

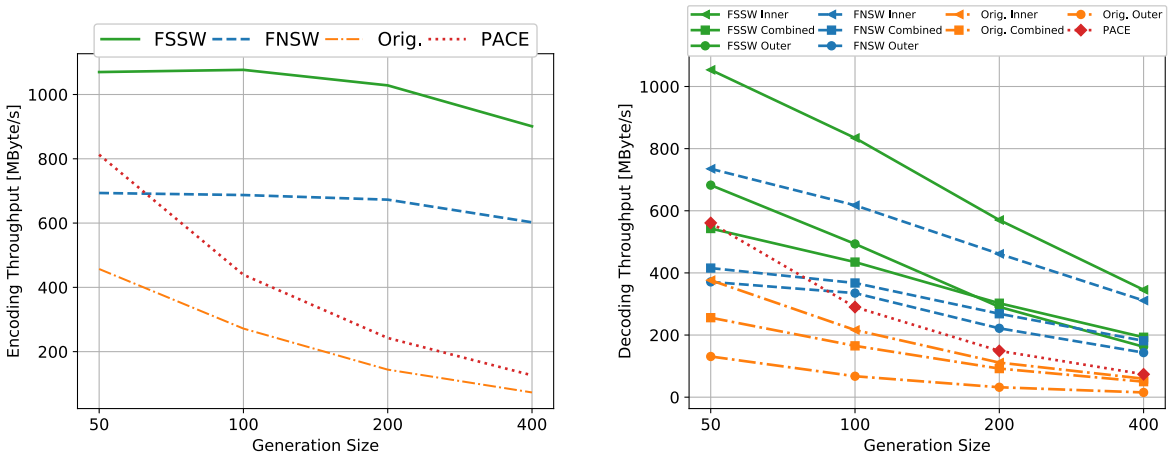
(d) Actual $\epsilon = 0.05$; estimated $\epsilon = 0.15$

FIGURE 4. In-order delay of individual packets $\delta(i)$ as a function of packet order number i for FNSW, FSSW, original Fulcrum, and PACE for scenarios with correctly estimated channel packet erasure probabilities $\epsilon = 0.05$ in (a) and $\epsilon = 0.15$ in (b), and incorrect estimation, namely estimated $\epsilon = 0.05$ and coding parameters from Table 3 for $\epsilon = 0.05$, but actual $\epsilon = 0.15$ in (c), while estimated $\epsilon = 0.15$ and coding parameters from Table 3 for $\epsilon = 0.15$; but actual $\epsilon = 0.05$ in (d); fixed parameters: generation size $N = 64$, $r = 4$ outer coded packets.

until dense coded packets arrive at the end of the generation, causing an underlying decreasing delay trend for increasing packet order number i, see the mild trend in Figure 4(a) and the pronounced trend in Figure 4(c). Whereas, too many coded packets delay the transmission of the next window, causing an underlying increasing delay trend, see Figure 4(d).

For FSSW, Figure 4 indicates that the in-order packet delays are consistently below 12 time slots for all considered scenarios. FSSW achieves the short in-order packet delays due to the transmission of the systematic packets. More specifically, a received systematic packet i can immediately be passed to the upper layer. If there are no more than c_{FSSW} packet erasures within a subset of u packets, then the erasures can be recovered by the c_{FSSW} coded packets at the end of the subset (provided these c_{FSSW} coded packets are linearly independent). If the packet erasure probability ϵ on the channel has been correctly estimated, then the c_{FSSW} coded packets are typically sufficient for the recovery, resulting in the nearly flat delay curves in Figures 4(a) and 4(b). On the other hand, if ϵ is underestimated and thus c_{FSSW} too small, then some erased packet can only be recovered with the dense coded packets sent at the end of the generation, resulting in the tendency for increased delays for the packets early in the generation and thus the decreasing delay trend with increasing packet number i in Figure 4(c).

A large number c_{FSSW} of coded packets at the end of a subset of u packets is helpful for ensuring the recovery of the erased packets of the subset; however, the c_{FSSW} coded packets delay the transmission of the subsequent systematic source packets by c_{FSSW} time slots. These dynamics lead to the four barely visible sawtooth waves in Figure 4(b), where the $c_{\text{FSSW}} = 3$ coded packets protect against the high erasure probability $\epsilon = 0.15$, while increasing the in-order delays of the packets at the beginning of the next subset. The downward trend within a given sawtooth (subset) is caused by the packet recovery with the coded packets at the end of the subset.



(a) Encoding (b) Decoding

FIGURE 5. Encoding and decoding throughput as a function of generation size N; fixed parameters packet erasure prob. $\epsilon = 0.1$, $r = 4$ outer coded packets, window size $w = 10$; FSSW: $u = 9$, $c_{\text{FSSW}} = 1$; FNSW: $m = 9$, $c_{\text{FNSW}} = 11$.

An excessive number c_{FSSW} of coded packets leads to the "step-up" curve of the in-order packet delays in Figure 4(d).

We also observe from Figure 4 that PACE with its structure of u systematic packets followed by c_{FSSW} dense $GF(2^8)$ coded packets achieves the same low delays as FSSW; albeit, PACE does not provide the flexibility of utilizing the inner, outer, or combined Fulcrum decoder.

C. IMPACTS OF GENERATION SIZE ON THROUGHPUT

This section examines the encoding and decoding (computation) throughput as a function of the generation size N. The channel erasure probability was set to $\epsilon = 0.1$. The window size was set to $w = 10$ packets so that FNSW can have a low coding complexity and low computation time and correspondingly high throughput levels. The effect of the window size w on the throughput will be examined Section III-D.

1) ENCODING THROUGHPUT

We observe from Figure 5(a) that FSSW and FNSW substantially increase the encoding throughput compared to the original generation-based Fulcrum, whereby the encoding throughput gap widens with increasing generation size N. In particular, FSSW more than doubles the original Fulcrum encoding throughput for $N = 50$; while for $N = 400$, FSSW achieves more than five times the original Fulcrum encoding throughput.

We also observe from Figure 5(a) that PACE achieves an encoding throughput between FSSW and FNSW for the small generation size $N = 50$, while the PACE encoding throughput drops to just slightly above the original Fulcrum encoding throughput for large N. PACE sends $u = 9$ systematic packets followed by $c_{\text{FSSW}} = 1$ dense $GF(2^8)$ coded packets that combine all previously transmitted source packets of the generation, whereas FSSW and FNSW utilize $GF(2)$ for the inner encoding and $GF(2^8)$ only for the outer encoding.

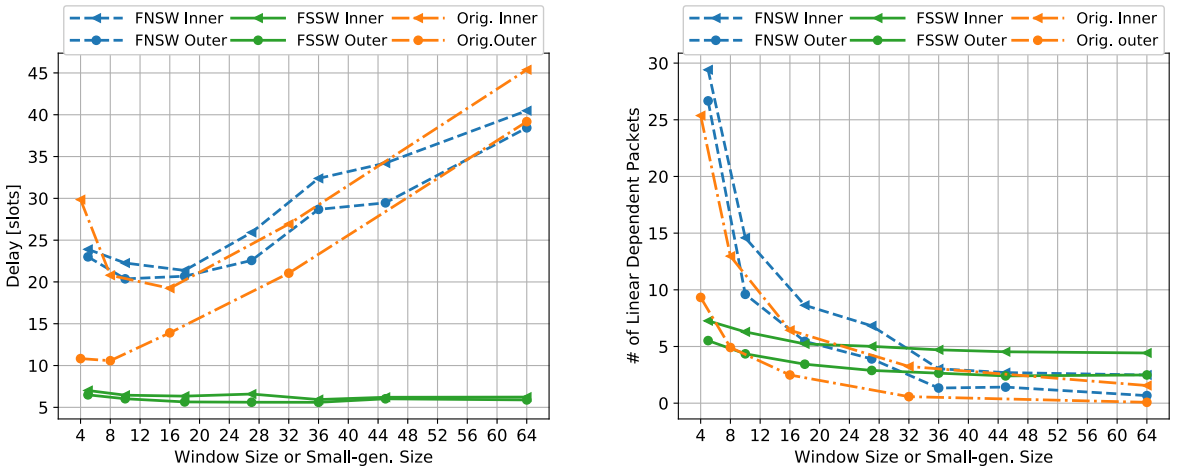
2) DECODING THROUGHPUT

Turning to the decoding throughput in Figure 5(b), we observe similar trends as for the encoding throughput. In comparison to the original Fulcrum, FNSW almost doubles the decoding throughput, while FSSW increases the decoding throughput approximately 2.5 times relative to the original Fulcrum. We observe from Figure 5(b) that for FSSW, the outer decoder achieves higher decoding throughput than the combined decoder for the small generation sizes $N = 50$ and 100; while for the large $N = 400$ generation size, the combined decoder achieves a higher throughput than the outer decoder, as is common for Fulcrum decoding [14]. The combined decoder applies two stages of inner decoding up to N received packets and then maps the resulting decoder coefficient matrix back to the $GF(2^8)$ outer decoding [14, Sec. II.D.3.]. In contrast, the outer decoder maps back to $GF(2^8)$ outer decoding right away. The large proportion of systematic source packets in FSSW simplifies the outer decoding (as the systematic packets are an extreme form of sparse coding [56]–[61]). Moreover, for small generation sizes N, the cubed computational complexity of decoding in N [62] is low. The additional computation overhead from executing the inner decoding stages of the combined decoder does not pay off for the combination of extremely sparse (FSSW) coding and small generations. On the other hand, for large generations of extremely sparse (FSSW) coding or any generation size of moderately sparse (FNSW) coding, the inner decoding stages of the combined decoder reduce the decoding complexity compared to pure outer decoding.

The PACE decoding throughput in Figure 5(b) follows a similar trend as the PACE encoding throughput in Figure 5(a) due to the exclusive operation of PACE in $GF(2^8)$.

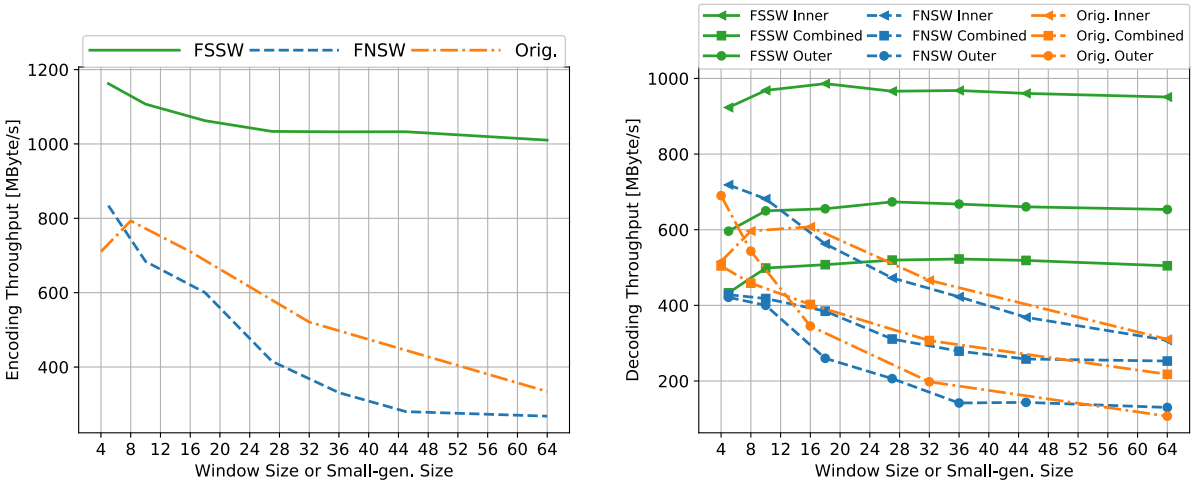
D. IMPACT OF WINDOW SIZE

This section examines the impact of the window size w on the mean packet delay D, the number of linear dependent coded



(a) Mean delay D (b) Number of lin. dependent coded pkts. per generation

FIGURE 6. Mean in-order packet delay D and number of received linear dependent coded packets per generation for FNSW and FSSW as a function of window size w with parameters from Table 4 and for small-generation original Fulcrum as a function of small-generation size η, see Section III-A4.



(a) Encoding (b) Decoding

FIGURE 7. Encoding and decoding throughput of FNSW and FSSW as a function of window size w for parameters from Table 4 and of small-generation original Fulcrum as a function of η, see Section III-A4.

TABLE 4. FSW parameters for generation of $N = 64$ packets, $\epsilon = 0.1$ packet erasure probability, and $r = 4$ outer coded packets for different window sizes w; FSSW code rate is set to $c = 1 - \epsilon = 9/10$ ($u = 9$, $c_{\text{FSSW}} = 1$, $t_{\text{FSSW}} = 76$) for all window sizes w.

Window	FNSW Parameters			
w	n_w	c_{FNSW}	m	t_{FNSW}
5	17	6	4	102
10	8	11	9	88
18	4	20	16	80
27	3	30	24	90
36	2	40	32	80
45	2	50	40	100
64	2	70	57	83

packets per generation, as well as the encoding and decoding throughput of FSSW and FNSW. Throughout this section, we consider the fixed generation size of $N = 64$ packets with $r = 4$ outer coded packets and a packet erasure probability of $\epsilon = 0.1$ on the channel. Table 4 summarizes the FSSW and FNSW parameters for the range of considered window sizes w. Figure 6 shows the mean in-order packet delay D and the number of received linear dependent coded packets per generation, while Figure 7 shows the encoding and decoding throughput as a function of the window size w.

1) MEAN IN-ORDER PACKET DELAY

We observe from Figure 6(a) that the FNSW mean in-order packet delay D generally increases as the window size w increases. This mean delay increase is mainly due to the sawtooth FNSW delay dynamics in Figure 4. As the window size w increases, there will be fewer, but larger sawtooth waves, which increase the mean value of the individual in-order packet delays $\delta(i)$. For the extreme case of $w = N + r$, there would be only a single sawtooth (window) and FNSW would degenerate to the original generation-based Fulcrum.

In contrast, we observe from Figure. 6(a) that the FSSW mean in-order packet delay D remains essentially unchanged as the window size w increases. There is only a very slight delay increase for the small window sizes $w = 5$ and 10 due to the slightly increased numbers of FSSW linear dependent packets for these small window sizes, see Figure 6(b).

We observe from Figure. 6(a) that the mean in-order packet delay D of small-generation original Fulcrum generally decreases as the small-generation size η is reduced; however, the delay D reaches a minimum for $\eta = 16$ for inner decoding and $\eta = 8$ for outer decoding and then increases for further reductions of the small-generation size η. In order to maintain the Fulcrum principle of at least one outer coded packet, the $\eta = 4$ small-generation coding requires a total of $N/\eta = 16$ outer coded packets per generation of $N = 64$ source packets, see Section III-A4. This large coding overhead negates the delay reduction effect of the small generation size. Moreover, the inner decoder cannot utilize the outer $GF(2^8)$ coded packets and requires the transmission of numerous inner coded packets to compensate for the high number of received linearly dependent inner coded packets that occur for small η, see Figure 6(b).

2) RECEIVED LINEARLY DEPENDENT CODED PACKETS

The number of received linearly dependent coded packets per generation in Figure 6(b) is evaluated by subtracting $N + r$ from the total number of packets received by the inner decoder (up to the point when enough packets have been received to decode all N source packets) and by subtracting N from the total number of packets received by the outer and combined decoder. We observe from Figure 6(b) that the number of linear dependent packets generally decreases with increasing window size w. This is because the number of possible random combinations in $GF(2)$ of the w source packets in the coding window increases as 2^w, reducing the probability of the occurrence of linear dependent coded packets. Specifically, the probability of receiving linearly independent packets in Eqn. (7) increases with w while c_{FNSW} and w satisfy Eqn. (3).

We further observe from Figure 6(b) that for a very small window size w (or small-generation size η), FNSW and original Fulcrum with inner decoding have more than double the number of linear dependent packets than FSSW and original Fulcrum with outer decoding. On the other hand, for large window sizes $w \geq 36$, FNSW and original Fulcrum have slightly lower numbers of linear dependent packets than FSSW. As illustrated in Figure 1(a), FNSW transmits all packets in inner $GF(2)$ coded form, while FSSW transmits only c_{FSSW} inner $GF(2)$ coded packets per u uncoded (systematic) source packets. For small window sizes w, and correspondingly few (2^w) possible random combinations of the source packets in the coding window, it becomes quite likely that FNSW generates linearly dependent coded packets. For large window sizes w (and correspondingly large c_{FNSW} that satisfy Eqn. (3)), the probability of receiving w linearly independent packets among c_{FNSW} coded packets increases, see Eqn. (7), and correspondingly the probability of linear independent packets increases for original Fulcrum with large values of the small-generation size η approaching the actual generation size N. On the other hand, for FSSW with large window sizes w, $w \gg u$, the first few coded packets in FSSW combine fewer than w packets; specifically, the first coded packet combines u source packets, as illustrated in the upper left corner of the coding coefficient matrix in Figure 1(b). Thus, for large window sizes w, $w \gg u$, FSSW slightly increases the chance of linear dependent packets compared to FNSW which always combines w source packets.

3) ENCODING THROUGHPUT

We observe from Figure 7(a) that FSSW achieves a high encoding throughput that decreases only slightly for increasing window size w; whereas, the FNSW encoding throughput decreases approximately quadratically with increasing window size w. FSSW generates only $\lceil c_{\text{FSSW}} \cdot (N + r)/u \rceil$ inner $GF(2)$ coded packets that are combinations of w coded packets, see Eqn. (10), leading to a linear increase in the encoding complexity with increasing w. In contrast, FNSW generates t_{FNSW}, $t_{\text{FNSW}} \geq N + r$, inner coded packets as specified in Eqn. (6) in conjunction with Eqns. (3)–(5). Each inner coded packet is a combination of w packets in $GF(2)$, and there are c_{FNSW}, $c_{\text{FNSW}} \geq w$ [see Eqn. (3)], coded packets per window, resulting in a computational encoding complexity that is quadratic in w [62].

Figure 7(a) also indicates that the encoding throughput of the small-generation original Fulcrum strategy is generally in the vicinity of the FNSW encoding throughput. The decrease of the original Fulcrum encoding throughput when decreasing the small-generation size from $\eta = 8$ to $\eta = 4$ is mainly due to the increase of the total number of outer $GF(2^8)$ coded packets from 8 to 16 for the full generation of $N = 64$ source packets, see Section III-A4.

4) DECODING THROUGHPUT

Figure 7(b) shows the decoding throughput levels for the three types of Fulcrum decoders. We observe from Figure 7(b) that for all decoder types, FSSW achieves higher decoding throughput than FNSW, mainly due to the systematic source packet transmissions in FSSW. We also observe from Figure 7(b) that the FSSW decoding throughput is nearly constant as the window size w increases; whereas, FNSW exhibits decreasing decoding throughput for increasing w. There are two main opposing effects are work: ($\text{TH}_{\text{Dec}} \searrow$) Larger window sizes w increase the computational effort for the decoding of the sparse coded packets, i.e., the packets that are combinations of w source (and outer coding) packets; and ($\text{TH}_{\text{Dec}} \nearrow$) larger window sizes w reduce the number of linearly dependent packets (see Figure 6(b)), thus reducing the need to decode dense coded packets (that are combinations of all $N + r$ source and outer coded packets) at the end of the generation. For FSSW with only few sparse coded packets, these two effects approximately compensate each other. On the other hand, for FNSW which requires

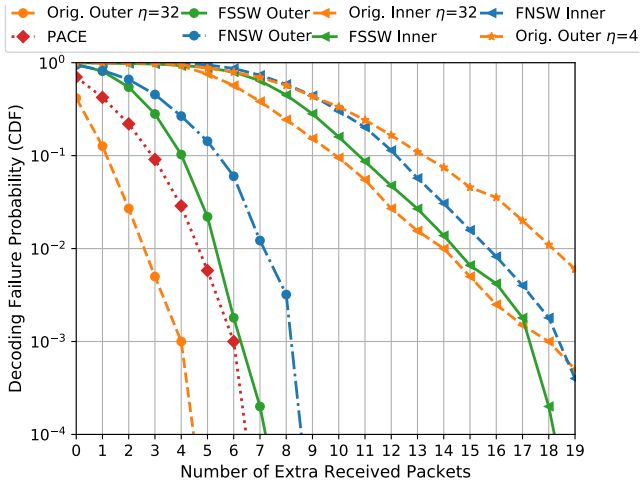


FIGURE 8. Decoding failure probability of $N = 64$ source packets as a function of number of received extra packets for inner and outer/combined Fulcrum decoder for FSSW, FNSW, PACE, and small-generation original Fulcrum; fixed parameters: packet erasure prob. $\epsilon = 0.1$, $r = 4$ outer coded packets, window size $w = 30$; FSSW: $u = 9$, $c_{\text{FSSW}} = 1$; FNSW: $m = 27$, $c_{\text{FNSW}} = 33$.

the decoding of blocks of $c_{\text{NFSW}} \geq w$ sparse coded packets (which involves computationally expensive matrix inversion with cubed complexity in c_{NFSW} [62]), the decoding throughput decreasing effect ($\text{TH}_{\text{Dec}} \searrow$) dominates for increasing w despite the pronounced drop in the FNSW number of linearly dependent packets with growing w (see Figure 6(b)).

Importantly, the throughput results in Figure 7 indicate that the FSW coding design provides a strong differentiation between the computational complexity of the different types of Fulcrum decoders. For the higher-performing FSSW, the inner decoder achieves about 3/2 times the throughput of the outer and combined decoders, i.e., inner decoding has only about 2/3 of the computational complexity of outer or combined decoding.

E. DECODING PROBABILITY

This section examines the decoding probability of the N source packets of a generation as a function of the number of extra received packets (beyond N packets) by the inner Fulcrum decoder and the outer/combined Fulcrum decoder. We observe from Figure 8 that small-generation original Fulcrum with $\eta = 32$, PACE, and FSSW with outer/combined decoder, as well as FNSW with outer/combined decoder require the fewest received packets, followed by the inner decoder for original Fulcrum with $\eta = 32$, FSSW, and FNSW. The original Fulcrum with $\eta = 4$ and outer decoding requires even more received packets. The outer/combined decoder maps back to the large-size $GF(2^8)$, where linear dependent coded packets and the omission of source packets in the outer coded packets due to a zero coding coefficient are very rare. Thus, in FSSW, the $r = 4$ outer coded packets can almost surely recover any erased systematically transmitted source packet. On the other hand, the inner decoder has to rely on the $c_{\text{FSSW}} = 1$ inner $GF(2)$ sliding window coded packet that follows after $u = 9$ systematic source packets and the inner $GF(2)$ dense coded packets that follow after the t_{FSSW} packet transmissions. The sliding window inner coded packets cover at most $w = 30$ source packets (while the dense coded packets cover all N source packets); in addition, linear dependent coded packets (as examined in Figure 6(b)) and source packet omissions in a $GF(2)$ coded packet (when the corresponding coding coefficient is zero) limit the recovery capabilities of the inner coded packets.

We note that generation-based Fulcrum (for the full generation size N) requires at least $N + r$ received packets for inner decoding [14]. In contrast, FSSW permits—in principle—inner decoding with N received packets: Suppose that the channel erases c_{FSSW} systematic packets from each subset of u systematic source packets and that the c_{FSSW} coded packets can recover these packet erasures (or none of the u source packets, but all following c_{FSSW} coded packets are erased). Then, all N source packets can be decoded (or are systematically received) from N packets received at the destination. FNSW transmits only coded packets, therefore the N source packets need to be obtained through outer decoding of the $N \times N$ coding coefficient matrix or inner decoding of the $(N + r) \times (N + r)$ coding coefficient matrix.

The PACE benchmark employs only $GF(2^8)$ coding, which has a negligible probability of linear dependent coded packets, thus reducing the number of required received packets compared to the FSW schemes that employ $GF(2)$ coding for all inner coding. The small-generation $\eta = 32$ original Fulcrum benchmark with outer decoding has the smallest number of linear dependent packets in Figure 6(b) and correspondingly requires the smallest number of extra received packets for decoding. However, for the small-generation size $\eta = 4$, original Fulcrum with outer decoding has a mean number of close to ten linear dependent packets per generation of $N = 64$ source packets in Figure 6(b). Accordingly, small-generation original Fulcrum with $\eta = 4$ requires around ten packets to approach successful decoding and over 18 extra packets are required to reduce the decoding failure probability below 1%, see Figure 8.

IV. CONCLUSION

We introduced Fulcrum Sliding Window (FSW) coding to reap the benefits of both the Fulcrum RLNC coding and the sliding window RLNC coding. Based on the Fulcrum coding, FSW coding flexibly reaches diverse sets of destinations that need to decode with low-complexity XOR operations in the small Galois Field $GF(2)$ or with computationally demanding operations in a large Galois Field, e.g., $GF(2^8)$. Based on the sliding window coding, FSW achieves low in-order packet latencies as well as high encoding and decoding (computation) throughput. More specifically, we developed Fulcrum Non-systematic Sliding Window (FNSW) coding and Fulcrum Systematic Sliding Window (FSSW).

Our extensive evaluations indicate that the introduced FSSW coding achieves short in-order packet delays in

conjunction with high encoding and decoding throughput. We also observed that FSSW preserves the Fulcrum RLNC feature of effective differentiation between the outer and combined decoding (high reliability, i.e., low decoding failure probability, at the expense of complex $GF(2^8)$ decoding) versus inner decoding (reduced reliability or slightly increased delay for low complexity $GF(2)$ decoding).

The PACE benchmark [25] achieves similarly short in-order packet delays and high reliability as FSW, but gives lower encoding and decoding throughput and, importantly, is limited to operating in a single Galois Field size. A small-generation strategy of the original generation-based Fulcrum RLNC coding [14] incurs a high coding overhead as the small-generation size is shrunken in an effort to achieve short in-order packet delays. The high overhead negates the delay reductions of the small-generation size. Also, small-generation Fulcrum coding creates a relatively high proportion of linear dependent coded packets, sharply reducing the reliability, i.e., increasing the decoding failure probability.

Future research can build on FSW in several directions. One interesting direction is to further accelerate the encoding and decoding through hardware acceleration modules [63], [64]. FSW allows the flexibility to support hardware modules that are limited to XOR operations as well as hardware modules that can execute operations in large Galois Fields. Another interesting future research direction is to explore deep learning techniques [65]–[68] to estimate the channel packet erasure probabilities and to adapt the coding parameters.

REFERENCES

[1] 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.
[2] E. Benamira, F. Merazka, and G. K. Kurt, "Dynamic optimized cooperation in multi-source multi-relay wireless networks with random linear network coding," *Electron. Lett.*, vol. 57, no. 8, pp. 347–350, Apr. 2021.
[3] N. Moghaddas-Gholian, V. Solouk, and H. Kalbkhani, "Relay selection and power allocation for energy-load efficient network-coded cooperative unicast D2D communications," *Peer Peer Netw. Appl.*, vol. 15, no. 2, pp. 1281–1293, Mar. 2022.
[4] S. Thieme, B. Schutz, and N. Aschenbruck, "Less is more: Choosing fair network coding parameters," in *Proc. IEEE 46th Conf. Local Comput. Netw. (LCN)*, Oct. 2021, pp. 123–131.
[5] 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.
[6] H. Alshaheen and H. Takruri-Rizk, "Energy saving and reliability for wireless body sensor networks (WBSN)," *IEEE Access*, vol. 6, pp. 16678–16695, 2018.
[7] 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, p. 184, Apr. 2021.
[8] A. A. M. Hassan, A. A. Qidan, M. K. Aljohani, T. E. H. El-Gorashi, and J. M. H. Elmirghani, "Random linear network coding in NOMA optical wireless networks," 2022, *arXiv:2202.04123*.
[9] E. Zeydan, Y. Turk, and B. B. Zorba, "Enhancing the capabilities of mobile backhaul: A user plane perspective," *J. Netw. Syst. Manage.*, vol. 30, no. 2, pp. 1–29, Apr. 2022.
[10] B. Bu, "Network coding data distribution technology between streams of emergency system based-wireless multi-hop network," *Comput. Commun.*, vol. 186, pp. 22–32, Mar. 2022.
[11] C.-H. Lee and H.-T. Li, "Preliminary analytics of video cache improvement using random linear network coding," in *Proc. IEEE Int. Conf. Consum. Electron.-Taiwan (ICCE-TW)*, Sep. 2021, pp. 1–2.
[12] 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.
[13] Z. Lin, Y. Wang, Y. Lin, L. Wu, and Z. Chen, "Analysis and optimization of RLNC-based cache placement in 5G D2D networks," *IEEE Access*, vol. 6, pp. 65179–65188, 2018.
[14] D. E. Lucani, M. V. Pedersen, D. Ruano, C. W. Sørensen, F. H. Fitzek, J. Heide, O. Geil, V. Nguyen, and M. Reisslein, "Fulcrum: Flexible network coding for heterogeneous devices," *IEEE Access*, vol. 6, pp. 77890–77910, 2018.
[15] H. Tang, R. Zheng, Z. Li, and Q. T. Sun, "Scalable network coding over embedded fields," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Jul. 2021, pp. 641–646.
[16] J. Choi, "Sliding network coding for URLLC," *IEEE Trans. Wireless Commun.*, early access, Dec. 2, 2021, doi: 10.1109/TWC.2021.3129996.
[17] 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.
[18] F. Karetsi and E. Papapetrou, "Lightweight network-coded ARQ: An approach for ultra-reliable low latency communication," *Comput. Commun.*, vol. 185, pp. 118–129, Mar. 2022.
[19] 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.
[20] 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.
[21] 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.
[22] 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.
[23] 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.
[24] D. E. Lucani, M. Medard, and M. Stojanovic, "Systematic network coding for time-division duplexing," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2010, pp. 2403–2407.
[25] 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.
[26] 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.
[27] R. J. Y. Enrique, G. Ismael, and M. Jose, "Fulcrum coding performance on fog computing architecture," *IEEE Latin Amer. Trans.*, vol. 18, no. 11, pp. 1966–1974, Nov. 2020.
[28] V. Nguyen, J. A. Cabrera, D. You, H. Salah, G. T. Nguyen, and F. H. P. Fitzek, "Advanced adaptive decoder using fulcrum network codes," *IEEE Access*, vol. 7, pp. 141648–141661, 2019.
[29] V. Nguyen, J. A. Cabrera, G. T. Nguyen, D. You, and F. H. P. Fitzek, "Versatile network codes: Energy consumption in heterogeneous IoT devices," *IEEE Access*, vol. 8, pp. 168219–168228, 2020.
[30] Y. Rivera, I. Gutierrez, J. Marquez, R. Porto, and S. Castano, "Performance dynamic coding RLNC LoRa on smart cities," in *Proc. IEEE CHILEAN Conf. Electr., Electron. Eng., Inf. Commun. Technol. (CHILECON)*, Dec. 2021, pp. 1–7.
[31] A. Cohen, M. Médard, and S. Shamai (Shitz), "Broadcast approach meets network coding for data streaming," 2022, *arXiv:2202.03018*.
[32] B. Li, X. Guo, J. Cong, and R. Zhang, "Sliding window based RNC scheme in UAV multicasting: Performance analysis and network optimization," *IEEE Internet Things J.*, early access, Dec. 8, 2021, doi: 10.1109/JIOT.2021.3133528.
[33] S.-R. Tong and C.-H. Yang, "Efficient broadcast scheduling at mobile cloud edges for supporting news-broadcast-on-demand over P2P streaming," *Peer Peer Netw. Appl.*, vol. 15, no. 3, pp. 1345–1356, May 2022.

[34] T.-K. Hung, S. K. Kaushal, and H.-F. Hsiao, "Content distribution network for streaming using multiple Galois fields," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2021, pp. 1–5.

[35] P. K. Mallikarjuna Shastry P. M, "Huffman coding and multi-generation mixing assisted network coding based MAC for QoS-centric secure data communication over MANETs," *Turkish J. Comput. Math. Educ. (TURCOMAT)*, vol. 12, no. 6, pp. 2146–2166, Apr. 2021.

[36] 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," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 4, pp. 1135–1149, Dec. 2021.

[37] 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," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 4, pp. 1150–1164, Dec. 2021.

[38] M. C. O. Bogino, P. Cataldi, M. Grangetto, E. Magli, and G. Olmo, "Sliding-window digital fountain codes for streaming of multimedia contents," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2007, pp. 3467–3470.

[39] P. Cataldi, M. Grangetto, T. Tillo, E. Magli, and G. Olmo, "Sliding-window raptor codes for efficient scalable wireless video broadcasting with unequal loss protection," *IEEE Trans. Image Process.*, vol. 19, no. 6, pp. 1491–1503, Jun. 2010.

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

[41] 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.

[42] B. Sun, C. Gui, Y. Song, and H. Chen, "Adaptive length sliding window-based network coding for energy efficient algorithm in MANETs," in *Proc. IFIP Int. Conf. Netw. Parallel Comput.*, Oct. 2017, pp. 13–23.

[43] Y. Xie, B. Sun, C. Gui, and Y. Song, "Performance analysis of variable length sliding window network coding in MANETs," in *Proc. 4th Int. Conf. Syst. Informat. (ICSAI)*, Nov. 2017, pp. 880–884.

[44] W. Feng, H. Luo, B. Sun, and C. Gui, "Sliding window-based network coding with cooperative communication in wireless networks," in *Proc. IEEE 2nd Adv. Inf. Technol., Electron. Autom. Control Conf. (IAEAC)*, Mar. 2017, pp. 605–609.

[45] 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.

[46] M. Karzand and D. J. Leith, "Low delay random linear coding over a stream," in *Proc. 52nd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2014, pp. 521–528.

[47] 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.

[48] 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. Cham, Switzerland: Springer, 2011, pp. 129–136.

[49] O. Trullols-Cruces, J. M. Barcelo-Ordinas, and M. Fiore, "Exact decoding probability under random linear network coding," *IEEE Commun. Lett.*, vol. 15, no. 1, pp. 67–69, Jan. 2011.

[50] J. Cloud and M. Médard, "Network coding over SATCOM: Lessons learned," in *Proc. Int. Conf. Wireless Satell. Syst.*, Cham, Switzerland: Springer, 2015, pp. 272–285.

[51] 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.

[52] 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.

[53] D. Malak, M. Medard, and E. M. Yeh, "Tiny codes for guaranteeable delay," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 4, pp. 809–825, Apr. 2019.

[54] W. Yang and Y. Li, "An irregular graph based network code for low-latency content distribution," *Sensors*, vol. 20, no. 15, p. 4334, Aug. 2020.

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

[56] 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.

[57] 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.

[58] 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.

[59] E. Tasdemir, M. Tomoskozi, J. A. Cabrera, F. Gabriel, D. You, F. H. P. Fitzek, and M. Reisslein, "SpaRec: Sparse systematic RLNC recoding in multi-hop networks," *IEEE Access*, vol. 9, pp. 168567–168586, 2021.

[60] 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.

[61] Q. Zhou, T. Zhao, X. Chen, Y. Zhong, and H. Luo, "A fault-tolerant transmission scheme in SDN-based industrial IoT (IIoT) over fiber-wireless networks," *Entropy*, vol. 24, no. 2, p. 157, Jan. 2022.

[62] 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.

[63] 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.

[64] 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.

[65] 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.

[66] H. Li, S. Yu, X. Lu, L. Xiao, and L.-C. Wang, "Drone-aided network coding for secure wireless communications: A reinforcement learning approach," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2021, pp. 1–6.

[67] Q.-T. Vien, T. T. Nguyen, and H. X. Nguyen, "Deep-NC: A secure image transmission using deep learning and network coding," *Signal Process., Image Commun.*, vol. 99, Nov. 2021, Art. no. 116490.

[68] Q. Wang, J. Liu, K. Jaffres-Runser, Y. Wang, C. He, C. Liu, and Y. Xu, "INCdeep: Intelligent network coding with deep reinforcement learning," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2021, pp. 1–10.

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, Technical University Dresden, Germany.

VU NGUYEN received the B.Tech. degree in IT from the Hue University of Education, the M.Eng. degree in computer science from the Da Nang University of Technology, Vietnam, in 2011, and the Ph.D. (Dr.-Ing.) degree in electrical and computer engineering from TU Dresden, Dresden, Germany, in 2021, specialized in network coding and fulcrum codes. He has been a Lecturer with the Vietnam-Korea University of Communication and Technology and The University of Danang, Danang City, Vietnam. He is currently a Senior Researcher with the Deutsche Telekom Chair of Communication Networks, TU Dresden, and a Senior Software Developer at Mimetik Company. His current research interests include network coding, high performance and low complexity communications, human–machine collaboration, and machine learning.

GIANG T. NGUYEN received the Ph.D. degree in computer science from TU Dresden, Germany, in 2016. He is currently an Assistant Professor, heading the Haptic Communication Systems Research Group, Faculty of Electrical and Computer Engineering, TU Dresden. His research interests include network softwarization and distributed systems.

FRANK H. P. FITZEK (Senior Member, IEEE) received the Diploma (Dipl.-Ing.) degree in electrical engineering from the University of Technology—Rheinisch-Westfälische Technische Hochschule (RWTH), Aachen, Germany, in 1997, the Ph.D. (Dr.-Ing.) in electrical engineering from Technical University Berlin, Germany, in 2002, and the Honorary (Doctor Honoris Causa) degree from the Budapest University of Technology and Economy (BUTE), in 2015. He is currently a Professor and the Head of the Deutsche Telekom Chair of Communication Networks, Technical University Dresden, Germany, coordinating the 5G Laboratory Germany. He is the Spokesman of the DFG Cluster of Excellence CeTI. He became an Adjunct Professor with the University of Ferrara, Italy, in 2002. In 2003, he joined Aalborg University as an Associate Professor and later became a Professor. He co-founded several start-up companies starting with Acticom GmbH in Berlin, in 1999. His current research interests include the areas of wireless and mobile 5G communication networks, mobile phone programming, network coding, cross layer, and energy efficient protocol design and cooperative networking. He was selected to receive the Nokia Champion Award several times in a row, from 2007 to 2011. In 2008, he was awarded the Nokia Achievement Award for his work on cooperative networks. In 2011, he received the SAPERE AUDE research grant from the Danish government and, 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 is also an Associate Editor of IEEE Access, IEEE Transactions on Education, IEEE Transactions on Mobile Computing, and IEEE Transactions on Network and Service Management. He currently serves as an Area Editor for *Optical Communications* of the IEEE Communications Surveys and Tutorials and as a Co-Editor-in-Chief of *Optical Switching and Networking*. He served as an Associate Editor of the IEEE/ACM Transaction on Networking (2009–2013), served as an Associate Editor-in-Chief of the IEEE Communications Surveys and Tutorials (2007–2020), and chaired the Steering Committee of the IEEE Transactions on Multimedia, from 2017 to 2019. He received the IEEE Communications Society Best Tutorial Paper Award, in 2008, and the Friedrich Wilhelm Bessel Research Award from the Alexander von Humboldt Foundation, in 2015.