IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 May 2023, accepted 11 July 2023, date of publication 20 July 2023, date of current version 28 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3297137

SURVEY

Network Coding as Enabler for Achieving URLLC Under TCP and UDP Environments: A Survey

PATRICK ENENCHE[1], **DONG HO KIM**[2], **(Senior Member, IEEE), AND DONGHO YOU**[1]

[1]Department of Information and Communication Engineering, Hannam University, Daejeon 34430, South Korea
[2]Department of Electronic and IT Media Engineering, Seoul National University of Science and Technology, Seoul 139743, South Korea

Corresponding author: Dongho You (dongho.you@hnu.kr)

This work was supported in part by Regional Innovation Strategy (RIS) through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (MOE) under Grant 2021RIS-004, and in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korea Government [Ministry of Science and ICT (MSIT)] [Development of the 6G Service Targeted AI/ML-Based Autonomous-Regulating Medium Access Control (6G STAR-MAC)] under Grant 2021-0-00368.

ABSTRACT In an era of rapidly evolving technology, the pursuit of higher data rates, extremely low latency, and robust reliability is becoming increasingly crucial. Applications such as Industrial Internet of Things (IIoT) demand that transmitted messages meet stringent time constraints. Traditional ARQ-based schemes struggle to achieve the required performance in the presence of feedback delays or feedback losses. Unlike conventional ARQ approaches, which acknowledge only original packets, Network coding (NC) can acknowledge multiple degrees of freedom (DOF), making the feedback mechanism a pivotal factor for reliability-delay tradeoffs. On the other hand, UDP protocol operates blindly without feedback acknowledgments (ACKs). Thus, the selection of code to provide ultra-reliable and low-latency communications (URLLCs) over lossy channels without feedback ACKs is tedious. However, network coding inherent robustness against packet erasure positions it as a promising candidate for reliable communication without feedback ACKs. In this survey, we delve into cutting-edge opportunistic network coding schemes that can deliver high data rates and low latency, even in the face of feedback limitations such as feedback delays, feedback losses, or complete feedback absence. We focus on how network coding-based approaches maintain effective communication over lossy networks, regardless of these feedback limitations. In the context of UDP environments (i.e., blind coding), we present the interplay of some essential elements that must be integrated into network coding to enable reliable transmission without feedback ACKs. While drawing support from secondary sources, our primary objective is to stimulate readers' interest in further exploration of network coding for URLLC in mission-critical (IIoT) applications.

INDEX TERMS Random linear network coding, URLLC, delay, reliability, ARQ, rateless code, feedback.

I. INTRODUCTION

In, 5G URLLC is an indispensable factor for countless applications especially in tactile internet, public safety, industrial automation, remote health care, and mission-critical systems (e.g. wearable computing devices and autonomous driving) [1], [2], [3], [4], [5], [6], [7], [8]. 5G and beyond will require an optimized data transport protocol to provide URLLC for the application of interest. In wireless packet networks, where packet loss remains a major issue leading to delays caused by channel randomness, the importance of an effective transport protocol cannot be overstated. Two commonly used protocols in this context are TCP and UDP, which are the focus of this paper. Under TCP environments, 5G must accommodate a round trip time (RTT) of approximately 1 millisecond, support the required overhead for sharing resources and access in 5G systems [9]. Thus, such critical time constraints constitute grave difficulties in setting up a protocol stack, packet scheduling technique and network core [10]. K-repetition strategy (i.e., repeatedly transmitting a packet over a random channel) and the adoption of forward error correction (FEC) codes can quickly help the decoder to

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

recover from packet loss due to the extra redundancy. Feedback packets (NACKs) are used to initiate the retransmission of original packets or additional redundant information generated through FEC codes. This process enhances transmission reliability in terms of packet delivery. Since, k-repetition strategy involves transmitting the same data packet K times in a row to enhance reliability, another responsibility of feedback is to minimize these repetitions. However, there have been difficulties in blending feedback and coding.

Achieving full capacity in terms of throughput and low in-order delivery (IoD) delay over an erasure channel is practicable for systems with perfect feedback, using automatic repeat request (ARQ) [11]. This performance also cuts across other links. However, when the channel becomes lossy and the feedback is delayed or lost, achieving this level of communication reliability is impossible. Simply put, as the channel becomes bursty, the ARQ performance degrades.

Combining ACKs and coding, an optimized ARQ scheme, where each received packet is treated as either useless or immediately decoded by the receiving device, has been proposed to improve the achievable rate [12]. In broadcast erasure channels with perfect feedback, decoding delay is optimized by the joint use of feedback and coding [13]. To minimize transmitter queue size, a network coding approach based on ARQ is presented in [11]. Contrary to the traditional ARQ scheme that acknowledges original packets, the approach in [11] takes advantage of the potential of network coding to acknowledge the degrees of freedom (DOF). This makes the feedback mechanism a determining factor for the throughput-decoding delay tradeoff [11]. Thus, in lossy channels associated with feedback delay and feedback loss, the scheme in [11] is relevant. Although several studies have considered systems with perfect feedback, few have considered delayed feedback and feedback loss-prone lossy channels.

Alternatively, the FEC mechanism can be implemented [14] without requiring a feedback protocol. In ARQ schemes, retransmission of original packets is made in response to receiver feedback. In contrast, HARQ schemes combine ARQ with FEC, where data transmission with FEC codes is initiated instead of retransmitting the original packets. UDP-implemented schemes, on the other hand, such as blind coding, apply FEC codes based on a prescribed code rate (R) given to the sender. For instance, assume that 5-bit long data is to be transmitted, and of these five bits, the last two bits are parity bits. If each bit requires a time slot for transmission, then 5-time slots are required to transmit three bits of actual data and two parity bits. Therefore, the code rate is $R = 3/5$. These FEC codes only involve the source and end nodes (i.e., source-based applications). Examples of FEC codes are the Reed-Solomon codes, turbo codes, low-density parity-check codes (LDPC), and digital fountain codes: LT codes [15] or Raptor codes [16]. Despite having certain drawbacks, these codes ensure reliable data transmission even in scenarios with high packet erasure or loss rates.

Furthermore, fountain codes sustain the advantage of generating an infinite number of encoded packets (based on k-repetition strategy) from n information symbols; thus, with at least n packets, the receiver can retrieve the message. It has low complexity in terms of encoding and decoding of the order $O(KlogK)$ (where the length of the input symbols is indicated by K) for LT codes [15], which is reduced to $O(K)$ by Raptor codes [16]. However, owing to their routing-based operation, optimal code performance is not guaranteed. As seen in Fig. 1(a), two transmission slots are needed to communicate with the robots. Thus, the transmission cost increases with increasing packet loss, as more slots are required. Thus, FEC (in this case, blind coding) can be used to improve the transmission success rate at the cost of multiple redundancies (e.g., using k-repetition strategy) to achieve low delay. Incorporating sliding windows, multipaths, and flexible codes in connection with shortcodes into network coding can help achieve URLLC.

Hence, because blind coding requires no feedback, it is essential to explore the elements of network coding and the interplay between various coding strategies to achieve URLLC without feedback. ARQ-based schemes are generally assumed to have delay-free (perfect and instantaneous) feedback [11], [17], [18]. However, this is never the case when the channel is prone to burst errors, resulting in delayed feedback, feedback loss, and RTT fluctuations. This drawback can affect the reliability of packet delivery. Presenting a survey paper on how network coding-based schemes are able to exercise their robustness towards these feedback limitations will be helpful for IIoT systems.

A. MOTIVATION AND CONTRIBUTIONS

Network coding, as discussed in [3] and [19], effectively reduces the number of transmissions, resulting in lower delays, higher throughput, and increased reliability. Its robustness is demonstrated by its capacity to recover the original packets at the destination, even when packet erasures occur. In contrast to forward error correction (FEC) codes [15], [16], network coding offers the advantage of fully utilizing bottleneck links. This is achieved by linearly combining each incoming packet at a node, which ultimately decreases the total number of transmissions [19]. As illustrated in Fig. 1(a), instead of separately sending packets d_1 and d_2, Fig. 1(b) demonstrates the transmission of their combination ($d_1 \oplus d_2$), maximizing bandwidth. By utilizing the ARQ approach, recovery of lost packets under TCP can be expedited, since a linear combination of previously sent but unacknowledged packets can be re-transmitted to compensate for the lost packets [20]. In this manner, the delay is minimized while increasing the throughput. Contrarily, in Fig. 1(b), under UDP, redundant NC coded packets ($m \times (d_1 \oplus d_2)$) can be sent to accompany initially sent packets to enable fast recovery [21], [22]. It is thus clear that multiple degrees of freedom (DOF) can be acknowledged

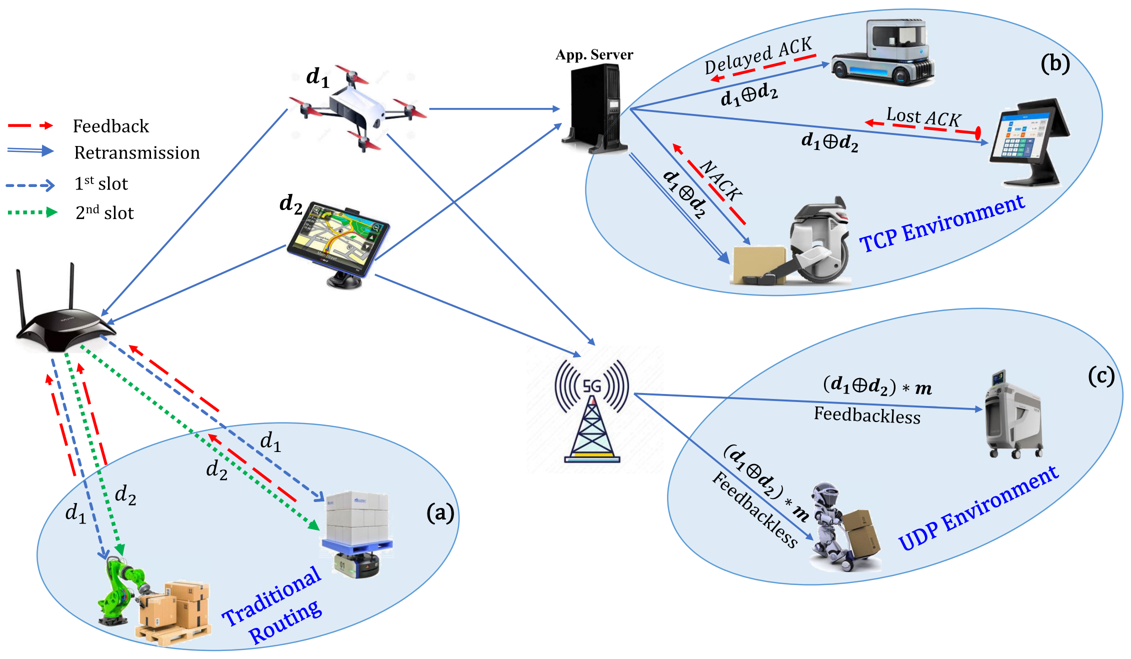


FIGURE 1. Illustrating NC as enabler in TCP and UDP environment for URLLC.

with network coding compared to traditional ARQ, as shown in Fig. 1(a).

We are motivated by the potential of network coding to address feedback limitations associated with ARQ schemes and its capability to achieve URLLC for specific applications without the need for feedback. In recent years, considerable progress has been made in demonstrating how network coding can ensure URLLC despite feedback challenges. The key contributions of this paper are as follows:

- We provided a brief network coding tutorial.
- We separately reviewed recent UDP and TCP related network coding schemes relevant for 5G URLLC and beyond.
- We classify network coding-based ARQ schemes for URLLC under TCP environments into two categories. The first category focuses on how network coding-based schemes achieve URLLC in a delayed-feedback-prone channel. The second category addresses how network coding-based schemes achieve URLLC in a feedback loss-prone channel.
- We provide how these schemes are able to adapt their way around these feedback limitations to achieve better performance.
- Likewise, under UDP environment, we provide how network coding strategies interplay to provide URLLC with no feedback at all. We refer to UDP as blind coding because it requires no feedback. Here we provide reviews on i) the interplay of sliding windows, short-codes, long codes, and k-repetition strategy in URLLC and how network coding improves performance, ii) the interplay of Sliding Window and Multipath/Multihop for URLLC, iii) the interplay of information size, overhead, and field size for payload delivery efficiency, and iv) Fulcrum network coding: The interplay of GF(2) and GF(2^8) or GF(2^{16}) for satisfactory communication towards achieving low latency.
- We also provide a brief summary of future research directions''

B. RELATED LITERATURES

To the best of our knowledge, this paper presents the first classification of network coding schemes for URLLC into categories based on their suitability for use with TCP (associated with delayed feedback and feedback loss channels) and UDP. Overall, some papers have provided reviews on ways to achieve low latency in 5G.

Among others, presented in [23] is an overview of leading 5G elements of cellular networks that can satisfy the low-latency requirements of 5G. The authors also provided three solutions to achieve low latency for emerging technologies. The Classification of various studies based on their approach towards proving end-to-end solutions to achieve low latency is presented in [24]. The approaches in wireless sensor networks (WSN) to achieve reliability and low-latency benchmarks in critical applications using the cross-layer quality of service (QoS) approach have been comprehensively reviewed [25].

In our evaluation of these related works, none of them zeroed in solely on network coding schemes; rather, some authors only considered it as one of the many approaches toward achieving low latency. Furthermore, none is categorized based on TCP (as it relates to delayed feedback and feedback loss) and UDP (which operates without feedback). Therefore, our survey presents details on how network coding-related schemes can adapt their way around these feedback limitations in lossy channels, to achieve better performance in terms of reliability and latency in 5G and beyond. In the context of IIoT, achieving URLLC is crucial to support mission-critical applications and time-sensitive processes. The ability to handle feedback loss effectively is vital for ensuring high reliability and low latency in the communication links between various IIoT devices, sensors, and actuators. Since IIoT networks often operate in harsh and noisy environments, it is crucial to explore methods to circumvent these feedback limitations to achieve URLLC.

The remainder of this paper is organized as follows: In section II, we present a brief background on network coding, a descriptive view of random linear network coding (RLNC) and its downside, and short introductory approaches to solving the problem. In light of feedback limitations, Section III expresses network coding schemes based on ARQ protocol for URLLC under TCP environment, while in Section IV, elements of network coding for URLLC under UDP environment is presented. In Section V, we provide a discussion for the study, including future perspectives based on our review, and finally, in Section VI, we draw a conclusion based on the content of our work.

II. NETWORK CODING: A BRIEF BACKGROUND

We cite an example to provide a clear but brief overview of network coding. Consider a router with incoming and outgoing links. It is important to note that a router's primary function is to store and forward packets received on the incoming links. This means that the outgoing links solely transmit copies of the packets received at the router without making any modifications or alterations. On the contrary, NC enables every node present in the network to compute a linear combination of the messages on the incoming link(s) such that the information on the node's outgoing link is a function of its input. At the destination node, the received data is then unmixed or decoded to recover the originally sent message.

The concept of NC was first introduced in [19] showing that it can increase network throughput, robustness to packet loss and link failures, reduction of network complexity as well as tighten network information security [26], [27], [28]. The benefit of network coding based on effective utilization of network resources has been studied [12], [29], [30]. The proceeding subsection demonstrates the benefit of network coding using the classical butterfly network.

A. CLASSICAL EXEMPLIFIED VIEW OF TRADITIONAL NETWORK CODING (THE BUTTERFLY SKETCH)

In this section, we depict with the help of a directed graph a generalized multicast network having a single source node and multiple receiving nodes. Specifically, as shown on the left (a) of Fig. 2, the source node (T_X) has an established connection with receivers R_{X1} and R_{X2}. Assuming that T_X wants

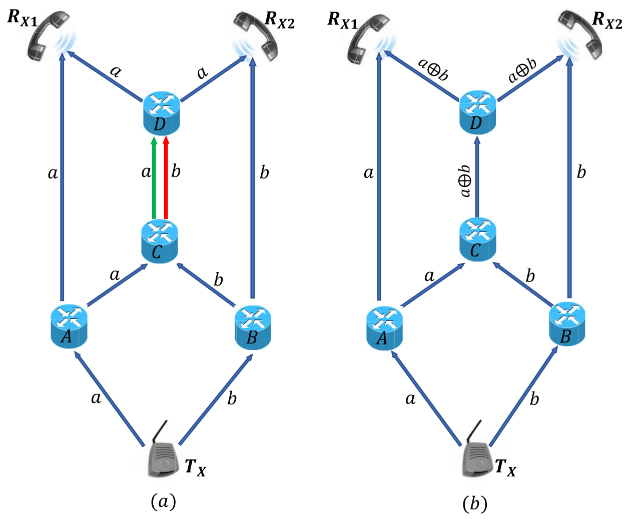
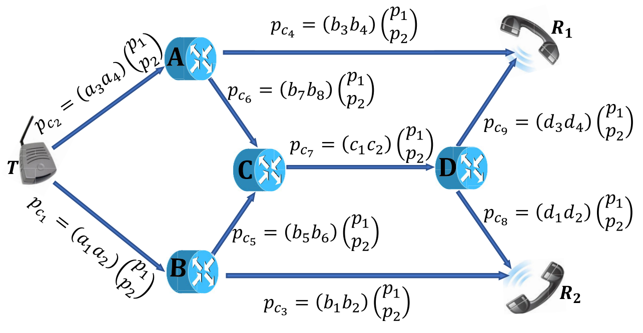


FIGURE 2. Traditional routing-based Butterfly network. The bottleneck link CD represented with green and red in (a) allows the flow of a packet for each time slot, *a* and *b* respectively. While (b) depicts a network coding-based butterfly network where the bottle link CD is fully maximized, as it allows for a linear combination of both packets [19].

to communicate two messages (a and b) to the destination nodes R_{X1} and R_{X2} at a rate 1 message per time slot, with traditional routing receiver R_{X1} and R_{X2} will successfully receive packet a and b respectively through paths A-R_{X1} and A-R_{X2} correspondingly. However, due to the limitation of traditional routing, it becomes impossible to simultaneously send both messages through the bottleneck link (CD) in a single time slot. Hence, 2 time slots (represented with green and red color) are required to transmit through link CD. So, R_{X1} receives its second packet via link A-C-D-R_{X1} in the first time slot while in the second slot, R_{X2} gets its second packet via path B-C-D-R_{X2}. Furthermore, R_{X1} receives packet a twice and R_{X2} likewise receives packet b twice, resulting in bandwidth wastage and redundancy. However, to fully maximize the bottleneck link CD as shown on the right (b) in Fig. 2, applying network coding, the incoming messages a and b arriving node C can be XORed giving rise to a new packet $c = a \oplus b$ on the outgoing link flowing towards node D which thereafter multicast c to final destination nodes R_{X1} and R_{X2}. R_{X1} can decode packet b by computing an XOR of a with c, clearly put, $a \oplus (a \oplus b) = b$. Similarly, R_{X2} obtains packet a by computing $b \oplus (a \oplus b)$. Using network coding, the following conclusion can be drawn; i) the time expended in the delivery of two packets has been cut down to a one-time slot, ii) bandwidth fully maximized, and iii) packet redundancy is reduced.

B. RANDOM LINEAR NETWORK CODING (RLNC)

Traditional network coding can be a complex and time-consuming process that requires optimal coefficient selection to combine different parts of a message into a single packet. However, vector network coding (VNC) [31], [32],



$p_{c_4} = (b_3 b_4)\binom{p_1}{p_2}$
R_1
$p_{c_2} = (a_3 a_4)\binom{p_1}{p_2}$
A
$p_{c_6} = (b_7 b_8)\binom{p_1}{p_2}$
$p_{c_9} = (d_3 d_4)\binom{p_1}{p_2}$
T
C
$p_{c_7} = (c_1 c_2)\binom{p_1}{p_2}$
D
$p_{c_1} = (a_1 a_2)\binom{p_1}{p_2}$
$p_{c_8} = (d_1 d_2)\binom{p_1}{p_2}$
$p_{c_5} = (b_5 b_6)\binom{p_1}{p_2}$
B
R_2
$p_{c_3} = (b_1 b_2)\binom{p_1}{p_2}$

FIGURE 3. **RLNC: Butterfly network [34].**

[33] or RLNC [34] simplifies this process by using random coefficients to mix message parts, making it faster and easier to implement. The vector of coefficients randomly drawn to generate the combination of packets can be placed in the packet headers to accompany the transmitted message. In Figure 3, RLNC is depicted, demonstrating that each node generates distinct coded packets, denoted as p_{c_i}, where i represents values ranging from 1 to 9. In this example, p_1 and p_2 are the packets being transmitted.

One of the major advantages of RLNC is its ability to recover lost or corrupted packets by decoding a linear combination of the received packets, making it more robust to packet loss than traditional network coding. Additionally, RLNC is highly scalable since it does not require a centralized controller or global knowledge of the network topology. Each node in the network can perform encoding and decoding operations independently, making it suitable for large-scale networks.

Security is also a significant advantage of RLNC. The randomly selected coding coefficients prevent eavesdropping and ensure confidentiality. As a result, an eavesdropper cannot infer the contents of the transmitted packets. Overall, RLNC offers several advantages over traditional network coding, making it an attractive choice for many network applications. Its simplicity, scalability, and security benefits have made it a popular choice for reliable communication over lossy channels.

C. TRADE-OFFS IN RLNC FOR ENHANCED URLLC PERFORMANCE: KEY FACTORS AND CHALLENGES

RLNC's ability to recover lost or corrupted packets through coding, without relying on explicit retransmission requests, makes it an efficient and flexible approach for data transmission in certain scenarios. However, the decoding process may introduce delays, affecting RLNC's suitability for time-sensitive applications. Optimal performance for these applications is achieved through progressive decoding of network-coded packets [35]. For clarification, during decoding there has to be at least a specific number of independent linear coded combinations of packets that is equal to the number of individually combined packets during transmission for successful decoding to occur [19], [36], [37], [38]. For instance, if a sender sends a total of $N = K + r$ packets where K is the exact number of packets needed for a successful message decoding at the receiver. While r represents the extra number of redundant packets to make up for any loss or erasure. Now in order for the receiver to start decoding without having hitches, there have to be at least K linearly independent coded packets. This requirement introduces decoding latency into the network which can be solved by employing a progressive decoding technique [35].

Using RLNC in network coding can increase the dimension of packets due to the combination of packets from one or multiple sources. However, this may increase the probability of decoding failure in the event of a node failure. Diversity-based network coding improves recovery via functional links but requires extra resources [39], [40] which comes at the cost of higher decoding complexity [41], [42].

Not only does RLNC introduce computational complexity, but it also brings about overhead issues. To illustrate, consider the analogy of sending letters as traditional data transmission, where each letter has a small address label. RLNC, on the other hand, combines multiple letters into a single package and includes a larger instruction sheet for reassembling them. These additional instructions (i.e., coding coefficients) contribute to increased overhead, potentially reducing delivery efficiency. This effect is particularly prominent in non-systematic RLNC, where even the original data packets require additional information, further amplifying the overhead.

On the other hand, applying a lower field size, such as GF(2), for coding restricts the degree of freedom in selecting coding coefficients. Consequently, the probability of forwarding multiple distinct symbols is lower. However, as coding is executed over a higher field size, GF(2^h) (where $h > 1$), the probability of sending several unique symbols over the network further increases. This increased probability reflects the enhanced degree of freedom provided by the larger field size, allowing for more diverse symbol combinations. Moreover, this probability corresponds to the linear dependency overhead, which, in turn, leads to delay. Therefore, both the field size and generation size exert a substantial impact on the resulting overhead [43].

For instance, high field sizes can lower the probability of linear dependence even though this can decrease the coding throughput [44], [45], [46], [47]. An attempt to reduce overhead using a smaller field size comes at the cost of linear dependency issues which translates into decreased decoding probability [48]. However, adopting smaller fields results in energy saving [43], [49], [50], [51]. Decreasing the message size – which includes overhead – reduces the degree of the probability of bit error, which in turn reduces communication delay. Efforts to improve decoding probability using higher field sizes have led to increased complexity [37]. Whereas, for large generation size transmission based on ARQ scheme, the feedback response is prolonged leading to a longer recovery delay [52]. Splitting the transmitted data size into smaller

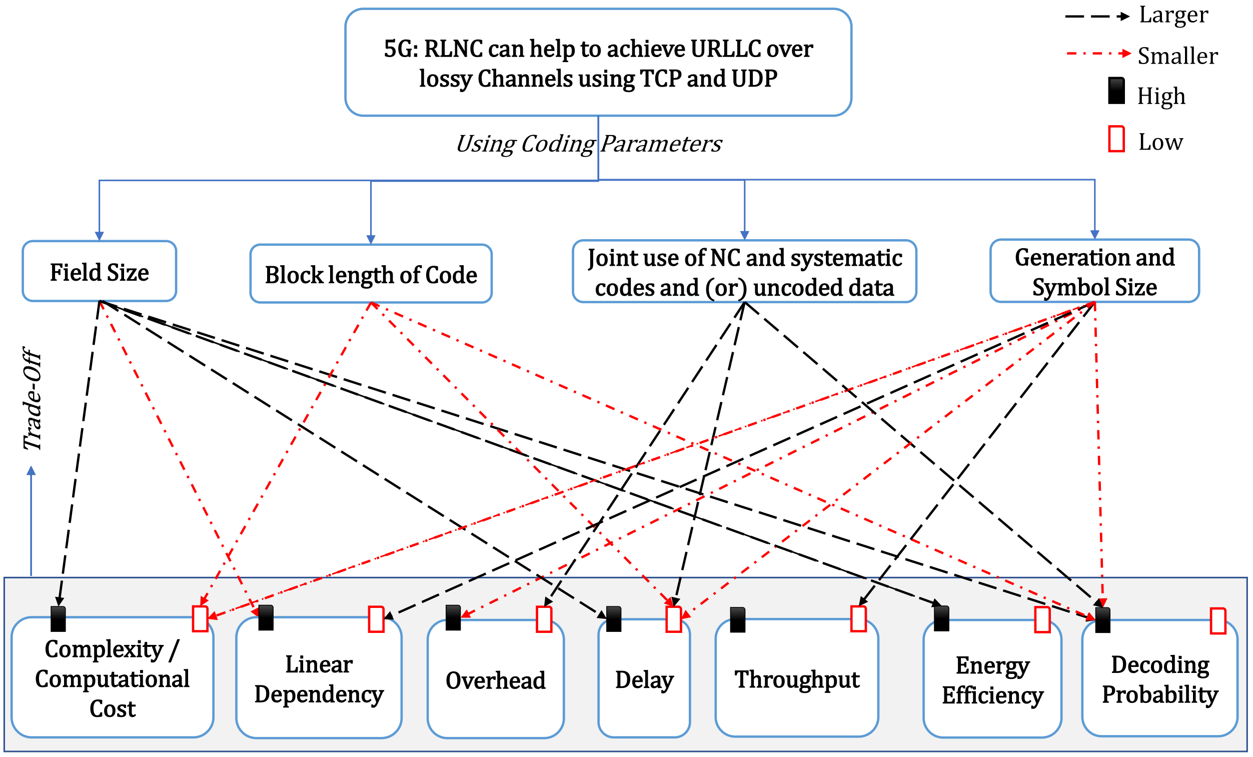

5G: RLNC can help to achieve URLLC over lossy Channels using TCP and UDP
Using Coding Parameters
Larger
Smaller
High
Low
Field Size
Block length of Code
Joint use of NC and systematic codes and (or) uncoded data
Generation and Symbol Size
Trade-Off
Complexity / Computational Cost
Linear Dependency
Overhead
Delay
Throughput
Energy Efficiency
Decoding Probability

FIGURE 4. **Drawbacks of RLNC.**

generation sizes can improve feedback response, leading to higher decoding probability and lower delay. However, this can increase feedback overhead, resulting in higher computational complexity [36], [46], [53], [54].

Feedback mechanisms in communication systems play a crucial role in enhancing reliability and reducing latency by allowing the sender to adjust its transmission strategy based on the network's current state. Detecting and responding to transmission errors or congestion in real-time helps deliver packets with high reliability and low latency. However, achieving URLLC without feedback is challenging, and while techniques like robust channel coding schemes, reducing packet size, and incorporating redundancy can help, they may not suffice for all URLLC applications. This highlights the importance of well-designed feedback mechanisms for optimal performance.

In the context of RLNC, achieving URLLC also depends on carefully considering various network code parameters, such as field size, channel utilization, adopted protocol, code selection, and generation or symbol size [55]. For clarity purposes, "generation" refers to the number of original packets in the transmission, while "symbol size" specifies the number of bytes in a single unit of data information. As shown in Fig. 4, these factors contribute to trade-offs among computational complexity, linear dependency/decoding delay, encoding latency, E2E delay, throughput, energy consumption, and overhead. Striking the right balance between these factors is essential for defining the required URLLC. As feedback mechanisms and RLNC both play a significant role in shaping URLLC performance, enhanced network coding schemes have been proposed to address their challenges. Extendable RLNC schemes can impact URLLC for 5G and beyond future communication systems, providing a foundation for more reliable and low-latency communication.

III. NETWORK CODING SCHEMES BASED ON ARQ FOR URLLC UNDER TCP ENVIRONMENT

Wireless networks typically use FEC and ARQ protocols to reliably send data over channels that may drop or corrupt some packets. However, these methods face various challenges due to factors such as feedback, real-time constraints, and congestion. Such challenges become even more problematic when the channel state fluctuates unpredictably, leading to variations in round-trip time (RTT) or bursts of lost data. Therefore, there is a need to employ an effective transport protocol to ensure safe delivery. In packet networks, any existing protocols for transporting data can be augmented by the application of NC. For instance, a transport protocol such as transport control protocol (TCP) can be influenced by Network coding to yield an optimized network [57].

A. NETWORK CODING MEETS ARQ PROTOCOL

Establishing reliable communication while transmitting via erasure channels over the years has become even more relevant with the emergence of packet networks. 1 - e represents the erasure channel's capacity, where e expresses the

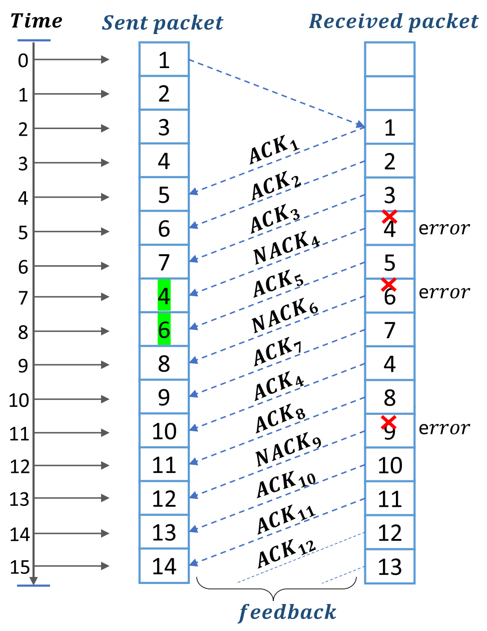


FIGURE 5. Selective repeat ARQ [56].

channel's erasure probability [58]. Attaining optimal channel capacity during transmission is usually a tedious task when the value of e is high. Fortunately, by the adoption of the famous ARQ protocol, achieving this capacity is an easy task for communication systems with strong feedback links. The process is straightforward yet effective, as the receiver only needs to send a message to confirm that each transmitted message has been received. If some messages are not acknowledged, they are retransmitted.

The extensive reliance of ARQ on feedback poses challenges for its implementation in practical systems, particularly when the feedback is delayed or lost. To address these challenges, Hybrid ARQ (HARQ) is adopted in both the former case [20], [59], [60], [61], [62], [63], [64], [65], [66] and the latter case [67], [68], [69], [70], [71]. HARQ is a combination of ARQ and FEC. These advanced methods incorporate an extra control code to mitigate errors by gradually transmitting more packets (symbols) until the message is successfully decoded. These systems have been proven to reduce the frequency of retransmissions. By reducing the frequency of retransmissions, these systems require fewer feedbacks which, in turn, plays a determining role in the tradeoff between reliability and delay. Additionally, they are designed to minimize energy consumption and address overhead issues, as reported in [56].

Selective Repeat ARQ (SR-ARQ) is a widely-used method of recovering lost packets in a communication channel. It works by retransmitting only the lost packets, in order to minimize the amount of data that needs to be resent. In the example shown in Figure 5, following the initial transmission of packets 1 to 7, any lost packets resulting in NACKs are selectively retransmitted to quicken recovery. Thus, only the 4th and 6th packets need to be retransmitted before transmission resumes with packets 8–14. However, this approach can still be slow if multiple packets are lost. An alternative Hybrid ARQ (HARQ) solution is to combine ARQ with RLNC. With this approach, a linear combination of packets can be transmitted in a single time slot, which speeds up recovery significantly. For instance, instead of retransmitting packets 4 and 6 separately within time slots 7 and 8 respectively, both can be sent together in a single time slot (i.e., 7). Additionally, network coding enables multiple degrees of freedom (DOF) to be acknowledged, giving greater flexibility compared to traditional ARQ.

When RLNC is used with feedback, RLNC is considered a type of Hybrid Automatic Repeat Request (HARQ), which combines error control coding with ARQ to improve reliability. However, if there is no feedback, RLNC cannot be considered HARQ, and decoding failures may result in the need for the receiver to request retransmission of the entire packet, which can lead to higher latency and lower efficiency. Thus, RLNC can improve upon the ARQ schemes and their variations by drastically reducing the feedback overhead. This can result in more efficient use of network resources and faster transmission of data, as less time and bandwidth are required for feedback.

B. RECENT NC-BASED ARQ SCHEMES FOR ESTABLISHING URLLC DESPITE FEEDBACK LIMITATIONS

Under this heading, we present recent network coding-based ARQ schemes that can maintain ultra-reliable low-latency communication (URLLC) even when there are limitations with the feedback mechanism. The limitations mentioned include delayed feedback and feedback loss, both of which can negatively impact reliability and latency in communication systems. Fig. 6 illustrates four instances of communication based on instantaneous feedback, delayed feedback, lost feedback, and no feedback (represented by f_i, f_d, f_l, and f_o respectively). The scenario assumes that receivers A, B, C, and D are all instances of communication with the base station (BS). In the scenario shown in Figure 6(a), the traditional ARQ protocol allows the base station (BS) to send packet p_2 to receiver B quickly and efficiently, as the BS receives instantaneous feedback (f_i) from receiver B. However, due to feedback limitations, the performance of the other receivers (A, C, and D) is likely to suffer. In contrast, with network coding, as seen in Fig. 6(b), BS is able to send a combination of both packets in a single time slot for all instances. However, except for instance B, to overcome the limitations of feedback, it is necessary to carefully design the coding decision complexities (δ, γ, and τ) for instances A, C, and D respectively. The following explores innovative network coding-based ARQ approaches that aim to achieve URLLC despite challenges such as feedback delay and feedback loss.

1) ACHIEVING URLLC DESPITE DELAYED FEEDBACK

To start with, we establish the concept of RTT relative to feedback. The concept of RTT is suggested for channel

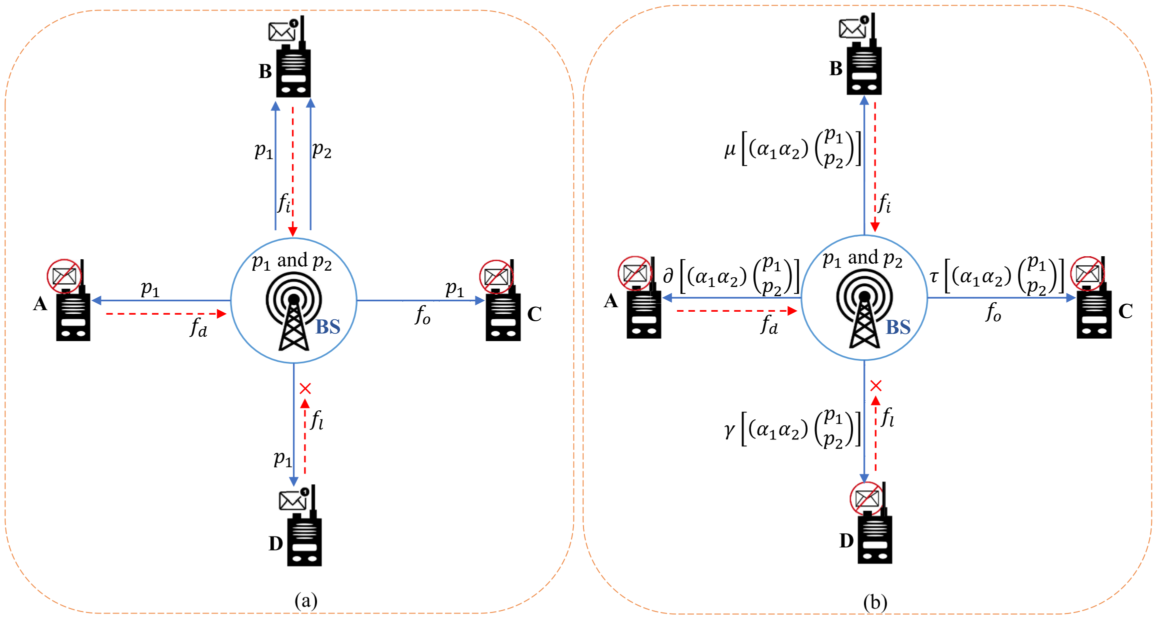

B
p_1
p_2
f_i
A
p_1
f_d
p_1 and p_2
BS
p_1
f_o
C
f_l
p_1
D
(a)
B
$\mu\left[(\alpha_1\alpha_2)\binom{p_1}{p_2}\right]$
f_i
A
$\partial\left[(\alpha_1\alpha_2)\binom{p_1}{p_2}\right]$
f_d
p_1 and p_2
BS
$\tau\left[(\alpha_1\alpha_2)\binom{p_1}{p_2}\right]$
f_o
C
$\gamma\left[(\alpha_1\alpha_2)\binom{p_1}{p_2}\right]$
f_l
D
(b)

FIGURE 6. Simple demonstration of (a) traditional ARQ and (b) network coding-based ARQ.

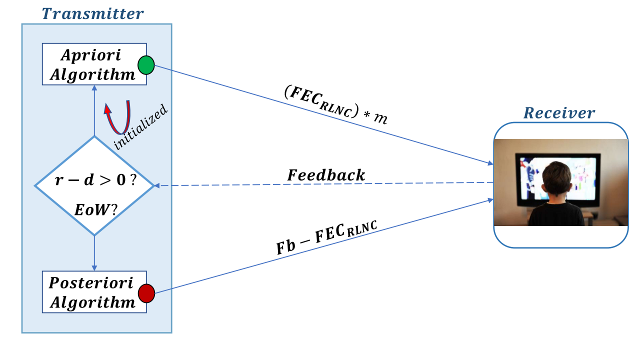


FIGURE 7. Description of the AC-RLNC model. Where d is the ratio of the degrees of freedom (Dof) needed for successful decoding to the transmitted number of Dofs contained in each RLNC coded packet c_t for a time slot t. EoW denotes the end of window.

monitoring [76], [77]. The RTT determines the promptness of the feedback. The use of ARQ in an instantaneous feedback environment has attracted extensive study. However, for systems with delayed feedback, the coding decision complexity increases even in the presence of slight feedback delays. In the following, we show how low latency is achieved despite feedback delay based on some coding decisions or approaches.

a: CAUSAL APPROACH BASED ON PRESCRIBED CODE RATE

To improve the performance of 5G technologies, HARQ error control protocols have been adopted [78]. However, these protocols work best when feedback is received without delay. When feedback is delayed, system performance can suffer, which can be problematic for time-sensitive applications. In addition, attempts to use SR-ARQ causes head-of-line blocking issues, leading to increased IoD delay of packets when feedback is delayed. In addition, the utilization of SR-ARQ can introduce head-of-line blocking issues, resulting in an increased IoD delay of packets when feedback is delayed.

Thus, the joint use of a systematic RLNC alongside a selective repeat ARQ (SR-ARQ) deterministic code has been proposed in [20] to hasten packet recovery from losses. To start with, the size of the information N to be transmitted is first split into k small generation sizes over which the coding window (CW) slides. Based on the time slots, the sender starts by sending uncoded packets in the first round. After the k^{th} packet is sent, it is followed by a network-coded packet containing the k packets in the CW. If the receiver is unable to decode due to erasure, it keeps requesting coded packets from the sender (based on the current window) until it has enough to successfully recover the lost packet(s). Then the window slides to the next generation of k packets and repeats the same process. The authors asserted that coding window size and redundancy as determinant factors for the IoD delay. The Matlab Simulation result showed that for some cases, the coding window size (generation size) that minimizes the expected IoD delay is moderately unresponsive to the network's RTT. We observed that when the loss probability (ϵ) is at most 10%, [20] still sustains an optimal performance. However, at $\epsilon = 20\%$, low delay is only possible at the expense of low throughput. This approach exhibits higher decoding complexity than [78], since it selectively retransmits lost packets in random linear network-coded form. Therefore, with an increasing number of retransmissions (if the channel becomes bursty), the message overhead will increase leading to decoding complexity.

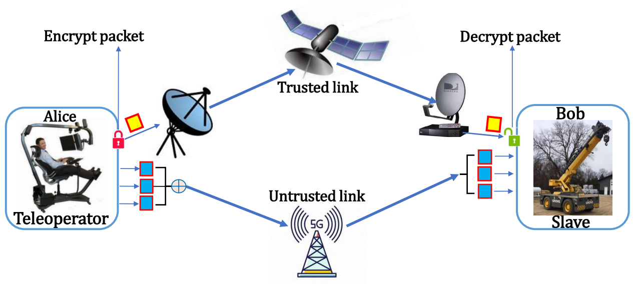


FIGURE 8. **Multipath approach for a secured URLLC.**

Contrary to retransmition of network-coded packets in response to the feedback (NACK) [20], retransmission of missing packets in uncoded (systematic) form has been reported [73]. The scheme is referred to as caterpillar RLNC with feedback (CRLNC-FB). When the sender receives a NACK, based on SR-ARQ similar to [20], it retransmits the missing packets in an uncoded manner to overcome burst error based on a prescribed code rate. The decoder, upon receiving these uncoded packets, simply passes them to the preceding higher protocol layer, on the condition that all the preceding symbols have already been decoded. As a result, the decoder does not have to wait to attain a full rank for the decoding process since the preceding symbols have all been recovered [79], [80]. Compared to [20], in terms of throughput-delay performance, CRLNC-FB provides a better result due to the incorporated sliding window technique in contrast with block-based RLNC approach in [20]. The use of a novel Band-form Gaussian elimination helps in the timely recovery of packets and since retransmitted packets are uncoded, CRLNC-FB decoding complexity will be less compared to [20]. For several conditions of various channels, by the adequate tuning (adapting) of the RLNC code rate, the desired throughput-delay trade-off can be obtained.

b: ADAPTIVE CAUSAL APPROACH BASED ON CHANNEL ESTIMATION

The causal scheme presented in [20] is only effective for random errors. While that of [73] can handle burst errors only to some extent because it maintains a prescribed code rate. However, the performance begins to degrade as the erasure rate of the channel becomes bursty, RTT starts fluctuating, or when real-time constraints are considered for transmission. This is because the coding is in a deterministic form. Therefore, [59] proposed an adaptive causal scheme to solve this problem.

In [59], the proposed adaptive and causal RLNC (AC-RLNC) algorithm is adaptive because it adjusts its code rates based on the current channel realization reflected by the feedback. It is causal because the algorithm is responsive to feedback. The sender begins the transmission by sending a coded combination of what is present in the current effective window as it expands until it gets to the end of the window (EoW) for k new packets. For instance, similar to [20], assume N is the information size and k is the effective window size, and $RTT = 4$. Then in the 1^{st}, 2^{nd}, and 3^{rd} time slots coded packets c_1, c_2, and c_3 will consist of packet(s) $\{p_1\}$, $\{p_1, p_2\}$, and $\{p_1, p_2, p_3\}$ respectively. At this point, the effective window is full (i.e. it has reached the end of the window (EoW)). Consequently, as shown in Fig. 7, the a priori algorithm based on the initial estimation of the channel rate (r) sends a RLNC FEC to accompany previously sent coded packets. Where m is the tunable parameter that determines the throughput-delay trade-off. The larger m gets the lesser the latency but the throughput declines when the redundancy becomes useless to the receiver. Secondly, After a complete RTT, based on the current channel status deduced from the received feedback (if it is a NACK), a feedback RLNC FEC (fb-FEC_{RLNC}) is retransmitted to adapt the transmission rate such that the expected throughput and delay are achieved by the receiver. We only explain the basic concept of this scheme here. However, to understand the essence of other conditional terminologies such as (d and EoW) expressed in Fig. 7, refer to [59]. The AC-RLNC also guarantees zero error probability in transmission. It achieves over 90% of the considered single-path (SP) channel capacity in a non-asymptotic regime. In line with commercial traces derived experimentally, throughput gains double, and mean IoD delay gain triples when the channel is in a bursty state compared to the SR-ARQ scheme in [20]. Only a small difference was recorded between the maximum and mean IoD delay in [59] compared to that of SR-ARQ [20]. However, since more coded packets are involved in AC-RLNC than in [20], the decoding complexity is more.

As we shall consider subsequently in [62], AC-RLNC has been investigated in multihop/multipath environment but for the sake of proper organization we skip it for now as this section consists of generalized approaches based on a point-to-point communication.

The challenge with the AC-RLNC despite its ability to lower the trade-off between throughput and delay is the inability of the sender to know the channel realization while the feedback is being delayed. Since short block codes are relevant for URLLC [81], [82], this gap can be reduced. Therefore, contrary to using long block codes as in [20], [59], and [62], short block length codes have been used to design a layered adaptive causal LAC-RLNC technique [60]. The LAC-RLNC divides the data it wants to transmit into f frames, which are base and enhancement layers respectively. The former is treated with a strict delay constraint while the latter is prioritized less. So, the sender transmits n coded packets over the network, consisting of k_1 packets of the base layer and the leftover $k_2 \leq n - k_1$ of layer 2. Since this approach employs the sliding window concept, the receiver can immediately start decoding the 1^{st} layer which is important for streaming to occur. Subsequently, 2^{nd} layer augments the base layer if it is decoded in the long run (as a result of delayed feedback). The throughput-delay trade-off is only managed by adapting the retransmissions of both layers based on their assigned priorities using the priori and posterior algorithms in [59]. Just like in previous schemes [20], [59], [62], zero error probability in transmission was recorded. Compared with AC-RLNC [59], a significant delay gain (by a factor of 3 for the base layer and 2 for the enhancement layer) in the mean and maximum IoD delay was achieved [60]. The base layer IoD delay was well near the optimal lower bound. While this scheme is able to raise the IoD delay gain of the base layer by a factor of 3.

c: ADAPTIVE CAUSAL APPROACH BASED ON DEEP-LEARNING

Adaptive and causal RLNC [60] though commendable as it is able to provide a balance between rate and in-order delay. However, predicting the channel state based on delayed feedback can be misleading if the prediction fails due to the dynamic nature of noise realizations. This inaccurate prediction can result in high in-order delay thus, degrading the system performance (affecting system causal behavior). Thus, an algorithm that predicts the channel status from data using the application of a deep learning-based approach called DeepNP is presented [83]. It is a statistical-based approach that predicts the noise realization to determine the state of the channel for AC-RLNC. It maximizes the throughput while simultaneously reducing the IoD delay for URLLC. Simulation results show a dramatic increase in the performance − mean and maximum delay gain rose by a factor of four and throughput by a factor of two − in contrast to the statistically-based AC-RLNC scheme [59]. Furthermore, even at 28% of the average error in prediction for each m transmission, authors claim this same performance can still be derived. The decoding complexity of [83] is expected to be much less than [20], [59], [60] since the amount of wasteful retransmissions due to inaccuracy in channel state prediction is lesser.

d: ADAPTIVE CAUSAL APPROACH BASED ON QUEUING THEORY

Inspired by queuing theory another coding decision problem has been addressed [63]. This inspiration stems from the fact that IoD delay is greatly impacted by the receiver queue occupancy (D_{qr}) than the transmitter queue. The authors derived a predictive scheduling protocol that enables the sender to predict the queue length of the receiver such that when the estimated queue length goes beyond a prescribed threshold due to unrepaired loss, a non-systematic (coded) packet is sent. Otherwise, the earliest native packet in the queue is transmitted. Finally, it does nothing if the transmitter queue is empty. Contrary to the sender deciding if a systematic coded packet [20], [59], [60], [83], systematic (uncoded) packet [73] or non-systematic packet should be transmitted based on the delayed feedback, in addition to these options, authors of [63] suggested that the node can also decide to do nothing. This added option helps to stabilize both receiver and sender queues, the link capacity is well treated, and lastly, at the receiver, the buffering delay is kept minimal. Results showed that with an increase in the feedback delay the conventional ARQ was outperformed with lesser delays. Also, less than a 5% loss in the capacity as a result of redundant packet transmission was observed. Furthermore, bounds were proposed for various corresponding queue lengths. Just like previous related approaches [59], [60], [83], the degree of prediction accuracy determines the trade-off between rate and delay.

In [63], the coding approach employed by the transmitter is adaptively designed based on random packet arrival and the estimated receiver queue length derived from the delayed feedback. However, the optimal coding decision as well depends on the size of the transmitter queue, because the total latency equals the sum of waiting packets present at both the transmitter and receiver. Hence, authors in [66] have factored into [63] and [75] the effect of geometric packet arrival at the transmitter and provide a partially observable markov decision process (POMD) that can achieve low E2E latency. Thus, [66] engaged the transmitter-receiver queue length in influencing the delay. By adjusting the POMD weighting parameter varieties of latency-channel utilization tradeoffs can be obtained. Where the ratio of the number of non-idle time slots (slots with transmission) to the total number of time slots defines the channel utilization. Simulation results show that at moderately larger channel utilization the scheme achieved low latency outperforming [63], [75]. The performance of ARQ was observed to drastically degrade with delayed feedback and growth in the complexity scheme was mentioned with feedback delay. Hence, the authors suggest that it be deployed in network scenarios with moderate feedback delay.

e: ADAPTIVE CAUSAL APPROACH BASED ON MULTIPATH

Innovative multipath TCP (MTCP) protocols can be used to establish a URLLC over a heterogeneous network. This concept finds application in connected autonomous vehicless (CAVs) [65]. Experimentally, some studies have shown that bulky transfers of files via links of identical quality results in a tremendous gain compared to traditional TCP [84], [85], [86]. However, MTCP is known to have some downsides as follows. i) Any missing packet must be figured out and retransmitted resulting in lengthy delays, and ii) also, the need for reordering of the transmitted packet (arriving from distinct links) at the sink can affect the efficiency of MTCP [84], [85], [86].

The reordering of the transmitted packets caused by head-of-line (HOL) blocking problem due to the difference in link qualities in a multipath setting has been overcome with the application of a discrete water-filling algorithm for multipath packet scheduling [61]. This approach reduces the transmission delay by scheduling the allocation of packets over multipath based on their different link qualities or erasure probability. Thus, if K DoFs are sent over Z paths, the performance of the system is determined by how much DoFs allocation is given to the bottleneck link alongside the number of transmissions it has to undergo. Results show that small K

guarantees minimal per-packet delay which is in agreement with [87] and [52]. Also, compared to the SP scenario, a significant gain was observed in terms of throughput and max per-packet delay.

With the application of [59], the impact of the AC-RLNC on first, a multipath network and secondly a combination multipath-multihop network topology has been studied [62]. For the former case, to maximize throughput gains while minimizing delay, a discrete bit-filling algorithm similar to the water-filling algorithm in [61] is used to adapt the allocation of packets for retransmission along each path. For the latter case, in order to minimize throughput degradation due to the differences in the channel quality of each hop, a decentralized balancing optimization algorithm is applied. In addition, at the intermediate node, a novel selective recoding is employed (i.e. not all packets traversing the intermediate node are recoded). Simulation results show that with zero error probability, the multipath case yielded the same channel capacity as in [59], [60], and [20] subject to average and minimal IoD delay restrictions. Whereas in the multipath-multihop case with respect to the achieved rate, the capacity is optimal. This approach yielded a throughput twice and delay reduction thrice that of SR-ARQ [20] and it also outperforms the SP scenario in [59]. However, the complexity tends to increase when the number of paths becomes high. Thus, at this point, the authors [62] proposed Knapsack problem algorithms [88], [89] to relax the optimization.

Furthermore, a scheme termed URLLC-MTCP is presented in [65] to overcome the prevalent reordering delay issues in connected autonomous vehicles (CAVs). In CAVs, the reordering delay issue is worse due to mobility, highly dynamic network topology, frequent handovers, fluctuating connections, etc. All these impede CAVs from satisfying the URLLC criteria in real-time [90], [91] Therefore, [65] undertook the following procedure to handle the problem i) streaming network codes were adopted for transmission such that MTCP transmitters can overcome reordering delays and link errors by increasing the amount of packet redundancy during transmission [92], [93], ii) Just like in [62], authors in [65] adopted a balanced link adaptation strategy to enable URLLC-MTCP to adapt its operation to the dynamically changing link status and past experience. This minimizes delay, maximizes throughput, and prevents overloading any single link, and iii) finally, a guaranteeable delay is reported within which any obtained delay should be. Results obtained show that every additional link caused a corresponding increase in the reliability of the connection and reduces delay especially when the network connections are unstable. With an increase in the channel error rate, a slow decline in performance was observed in the mean and guaranteeable delay of uncoded and coded URLLC-MPTCP. Notwithstanding, the coded URLLC-MPTCP showed a better performance. Compared to existing protocols of MPTCP, URLLC-MPTCP [65] is able to monitor the performance of any subflows over a period and informs the MPTCP-sender to remove any poorly performing link. Thus, URLLC-MPTCP can efficiently support delay-sensitive applications. However, its ability to dynamically adapt to the link quality incurs a penalty (Algorithm Complexity) which the authors claimed will be insignificant considering the advantage URLLC-MPTCP provides as the number of paths increases.

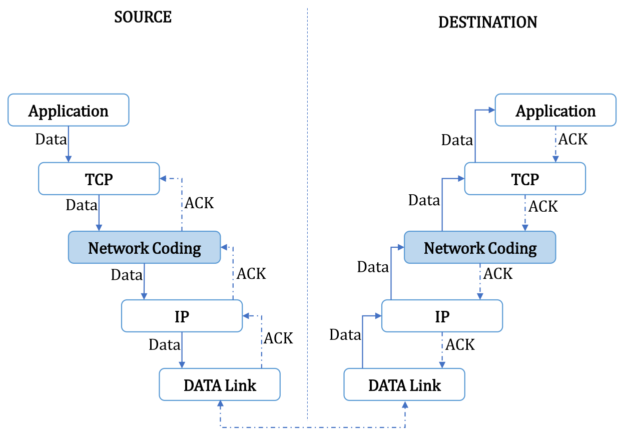


FIGURE 9. Network Coding layer grafted into the TCP/IP model.

Since transmission reliability can be altered in the absence of strong security measures, a scheme that achieves URLLC in the presence of an eavesdropper that may want to track and possibly corrupt traversing data in the network has been presented [64]. Termed as low-latency HUNCC, a multipath approach was employed to realize secured communication. As represented in Fig. 8, only a subset of the entire information represented by the yellow packet is encrypted and sent via the trusted link – in this case, a satellite link – while the remainder (blue packets) is transmitted through the 5G (untrusted link). Linearly mixing the packets using HUNCC as in [94] by the application of a post-quantum McElice encryption [95], ensures the security of the encrypted subset of the packet. On the other hand, applying AC-RLNC [59] on the 5G link helps to provide the trade-off between throughput and IoD delay. Thus, a combination of these two approaches gives a trade-off between security, throughput, and in-order delivery delay. Results showed that adopting these two distinct links for secured URLLC outperforms SP SR-ARQ [20], SP AC-RLNC [59] and multipath multihop AC-RLNC [62] schemes. Table 1 gives a summary of the pieces of literature we reviewed on schemes for achieving URLLC despite delayed feedback.

2) ACHIEVING URLLC DESPITE FEEDBACK LOSS

Bidirectional loss as illustrated in Fig. 6(b) instance D, is said to occur when there is packet loss in both forward and feedback channels. To better understand the information presented in Fig. 6(b), please refer to Section IIIB, paragraph one. Previous TCP schemes [76], [77], [96], [97], [98] have been unable to handle packet loss in heavy non-congestion loss networks. Consequently, the goodput rate decreases thereby causing the system reliability to drop. Therefore, TCP

TABLE 1. Achieving URLLC despite delayed Feedback: Approaches and Attributes.

Reference	Approach	Main Finding	Performance Metric
[20]	Based on a SR-ARQ, a deterministic systematic coding approach is employed to reduce IoD delay. After sending the k^{th} uncoded packets based on the prescribed code rate, a random linear coded packet is transmitted to hasten packet recovery. For every NACK received, a coded packet is retransmitted	The coding window size and redundancy are determinant factors for minimizing IoD delay. Its performance decreases as the channel becomes bursty RTT	in-order delivery delay
[73]	The same approach used in [20] is adopted to IoD delay. The only exception is that instead of retransmitting network-coded packets in response to a NACK, packets are resent in uncoded form	It Performs better in comparison with [20] and the computational complexity is less due to the retransmission of packets in uncoded form. Handles burst errors only to a small extent	Throughput-delay trade-off
[59]	Proposed an adaptive and causal systematic network coding scheme that adapts its coding parameters based on the current channel status deduced from the delayed feedback. This approach uses a dynamic code rate unlike [20], [74]	Expected throughput and delay are achieved as the mean IoD delay gain tripled compared to that of the SR-ARQ [20]. Also outperforms [74]	Throughput-delay trade-off
[60]	Using short block codes, scheme [60] is optimized to improve the throughput-delay trade-off for data streaming. More priority was assigned to the base layer while the other was treated less	Compared to [60] the IoD delay gain was raised by a factor of 3 and 2 for the base and enhancement layer respectively	Throughput-delay trade-off
[83]	Using the same framework in [60], the authors present an algorithm called DeepNP that predicts the channel status based on the noise realization estimated from the delayed feedback using a deep learning-based approach for URLLC	Simulation outcome shows a dramatic rise in mean and maximum delay gain by a factor of four and throughput by a factor of two in contrast to [60]	Throughput-delay trade-off
[63]	This adaptive and causal scheme predicts the channel status based on queuing theory approach. Thus, the sender can from the feedback determine the queue length of the receiver and then adapt the coding parameter	Similar to [60], [61], [84], the degree of prediction accuracy determines the trade-off between rate and delay	packet rate and IoD delay
[66]	This optimizes [64], [76] transmitter-receiver queue length to adapt the coding parameter using a POMD to provide low E2E latency	Gave better performance than [64], [76]. Also, diverse trade-offs can be derived between channel utilization and latency by fine-tuning POMD weighting coefficient	Channel utilization vs latency trade-off
[61]	Proposed a discrete water-filling algorithm for multipath. Based on the various estimation of each link rate, the algorithm minimizes the IoD delay by allocating more DoFs of coded packets to links with low packet loss probability	Small information size guarantees minimal per-packet delay. Also, compared to the SP scenario, a significant gain was observed. Performance is affected by the most delayed prone path	Throughput and per-packet delay
[62]	Proposed a discrete bit-filling and a decentralized link balancing algorithm that adapts the allocation of packets for retransmission and balancing throughput respectively in a multihop-multipath network	Simulation outcome showed throughput 2 times and delay 3 times that of the SR-ARQ	Throughput-delay trade-off
[65]	Using streaming codes for overcoming reordering issues, and a balanced adaptation strategy that enables multipath to dynamically adapt to its changing link status is proposed	Every additional link led to a corresponding rise in the reliability of the connection while achieving low latency	Reliability and latency
[64]	To provide secured URLLC, only a subset of the entire information is encrypted and sent via the trusted link (satellite link) while the remainder is transmitted through the untrusted link (5G) using low-latency HUNCC and AC-RLNC respectively	A combination of these two approaches gives a trade-off between security, throughput, and in-order delivery delay	Throughput, security, and latency

and network coding called TCP with NC (TCP/NC) have been proposed [67]. This concept led to the insertion of an additional layer between TCP and the IP layer, as illustrated in Fig.9. Basically, a combination of $n + K$ network-coded packets is generated from n original packets and sent over the network. On the receiver side, if only n coded packets are received, successful decoding can still occur despite the loss of k packets. This performance is only possible when loss happens only in the direction the data is being sent. In a bidirectional loss environment, performance in terms of delay and reliability is affected. Thus, TCP/NC cannot guarantee good performance in a feedback loss-prone channel. This is

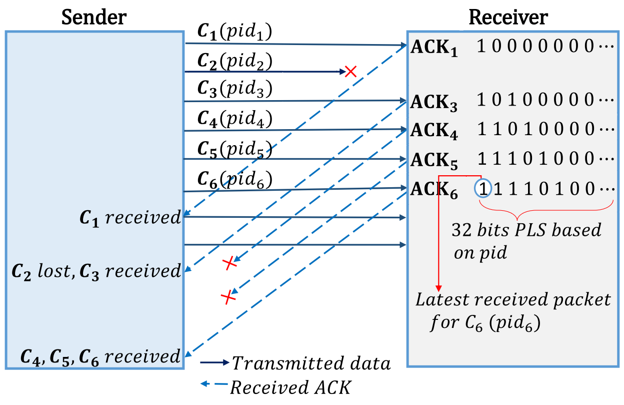


FIGURE 10. 32-bit packet loss sequence based on pid for system recovery from feedback loss [69].

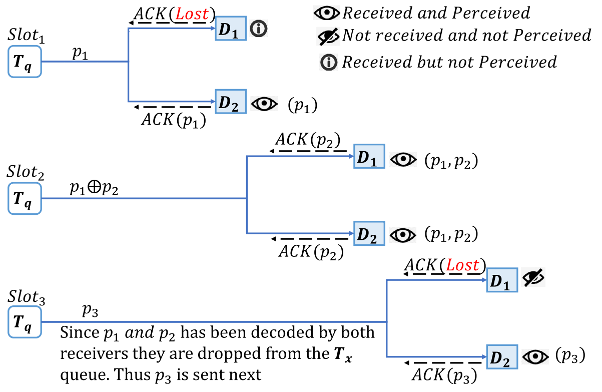


FIGURE 11. Upper triangular Generation matrix model for continuous innovative transmission.

because the channel will fail to update its coding parameters because it depends on feedback.

Thus, a special scheme TCP/NC that allows for link Loss-rate and Loss-burstiness Estimation, called TCP/NCwLRLBE has been proposed [68]. In this scheme, the authors introduced a novel information field (known as packet identifier (*Pid*), and *Pid-Echo-Reply*) to the network coding acknowledgment (NC-ACK) header. The former enables the sender to distinguish the ACKs of each coded packet combination from the others. While through the *Pid-Echo-Reply*, the sender can know the lost packets. As a result, the sender also can estimate channel burstiness. Notwithstanding, the scheme was insufficient in handling densely bursty feedback loss which translates to decreased performance in terms of delay and reliability.

In [99], TCP/NC has been deployed in a bidirectional loss environment. In addition to the newly introduced information field to the NC-ACK in [68], the authors added an 8-bit packet loss sequence (PLS). This PLS keeps records of ACKs from the oldest to the latest combination. So, every latest ACK carries the ACKs of previous combinations. Again, this approach cannot survive a dense bursty feedback loss. Thus, a new approach called bidirectional tolerance TCP/NC comes into play for dense bursty error correction [69].

As reported in [69], the scheme is able to maintain a goodput in an error-prone channel by adopting these proceeding techniques; i) acknowledgment (ACK) accumulation: borrowing leave from [99] and [68], a 32-bit PLS field is added to the NC-ACK header. As shown in Fig. 10, the PLS is signified by bits of $0's$ and $1's$, standing for lost or received respectively. The most recent 32-packet combinations from the oldest (rightmost bit) to the latest (leftmost bit) received successful combinations can be figured out by every latest ACK. So, even in the event of feedback erasure (ACK_4 and ACK_5), the sender can still retrieve sufficient information to estimate the channel status – with just ACK_6, the sender knows C_4, C_5, and C_6 have been received. However, additional negligible overhead is incurred on the ACK from 48 bytes [99] to 52 bytes due to the included PLS field, ii) ACK retransmission: to avoid TCP time out (TCP TO) after prolong waiting by the sender resulting from lost ACK, a retransmit of new ACK every 200*ms* is enabled. Thus preventing the current congestion window (CWND) size from resetting itself to 1, and iii) Duplicate generator: the sender by itself generates substitutes of the lost duplicate ACKs (provided source is aware of all lost ACKs number) relative to the PLS field and then delivered to the upper layer. These three innovations make TCP/NCs bidirectionally loss tolerance. Hence, it is called TCP/NCwBLT.

While the goodput performance of TCP/NCwLRLBE [68] decreases with increased packet loss rate due to numerous retransmissions and redundancy, simulation results show that the goodput of TCP/NCwBLT [69] is maintained irrespective of the Ack loss rate [68]. Contrarily, the TCP/NCs model [67] quickly experiences a decrease in goodput due to the congestion control algorithm it uses that always leads to the wrong decision of resetting the CWND. The goal of all these approaches is to reduce the number of wasteful retransmissions as much as possible while adapting to the dynamic channel status. Thus, TCP/NCwBLT can be a good fit for achieving URLLC. Although this approach seems more complex than [68], [99].

A different approach has been investigated in [70]. The scheme is termed upper triangular NC (UTNC). It works with a generation matrix (G) to handle the problem of wasteful retransmission of packets to reduce the decoding latency. The authors carried out the investigation using a broadcast approach to serve n receivers. As represented in equation (1), assuming the packets to be transmitted are $p_1, p_2, p_3, \ldots, p_m$ (meaning m original packets are present), thus we can have M sets of $p_i \;:\; i \in \{1, 2, 3, ..m\}$. The G matrix stores all the M sets of the packets. At every time slot, only linear combinations of the non-zero elements from the G matrix are transmitted. Since the transmission queue (T_q) houses all the original packets in a proper sequence, once any packet is sensed by all receivers or when p_i's decoding delay exceeds the decoding time constraint, it is ejected from T_q. Hence, it is termed the 'drop-when-perceived algorithm'. Find the model architecture in Fig. 11.

$$G = \begin{bmatrix} G_{1,1} & G_{1,2} & G_{1,3} & \cdots & G_{1,m} \\ 0 & G_{2,2} & G_{2,3} & \cdots & G_{2,m} \\ 0 & 0 & G_{3,3} & \cdots & G_{3,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & G_{m,m} \end{bmatrix}. \quad (1)$$

Therefore, with UTNC, despite the possible occurrence of feedback erasure, at every time slot, an innovative coded packet is always transmitted. To instantiate, lets say in the 1st time slot set $S_1 = \{p_1\}$ of $G_{1,1}$ was broadcast to destinations D_1 and D_2. From the feedback obtained, assuming p_1 was received by both D_1 and D_2 but D_1's feedback was lost (i.e. only D_2 perceived p_1 but D_1 did not base on UTNC algorithm). Instead of the sender retransmitting only p_1 in the second time slot [11] leading to poor channel utilization and wasteful redundancy, coded packets from the next set $S_2 = \{p_1, p_2\}$ of $G_{1,2}$ is transmitted. Since the coded packet is innovative, it is likely D_1 and D_2 may perceive p_1 and p_2. If so, D_1 and D_2 now has p_1 and p_2. Consequently, they are ejected from the T_q, and then the next set $S_3 = \{p_3\}$ of $G_{1,3}$ is transmitted and the process iterates through. Simulation results revealed that based on decoding speed and eradication of meaningless retransmissions introduced by feedback errors, the throughput increased in contrast with the previous scheme [11] where retransmission of lost packets has to be separately undertaken, thus degrading system performance. However, the drawback with UTNC is that the throughput tends to decline with a decrease in the decoding delay constraint. The future direction in line with the UTNC is to fine-tune its code configuration possibly with the aid of a sliding window to improve the latency for reliable communication for time-sensitive systems. Table 2 gives a summary of the literature we reviewed on schemes for achieving URLLC despite feedback loss.

IV. ELEMENTS OF NETWORK CODING FOR URLLC UNDER UDP ENVIRONMENT

One major challenge of data transmission without any feedback mechanism is the design of codes that can effectively combat packet loss. In order to accomplish these distinctive designs, it is crucial to consider specific elements of network coding. We will initiate this section by first exploring the concept of ratelessness, followed by an examination of its relationship with network coding.

A. RATELESS CODES MEETS NETWORK CODING

Existing communication systems use re-transmission to combat the degrading effect of erasure channels, but this method relies on feedback which may be unavailable. In contrast, FEC does not require retransmission but is prone to large errors when transmitting large data. Applying the ARQ protocol can lead to high power consumption due to the need for multiple retransmissions, which can increase overhead. Furthermore, correct decoding becomes impossible in the

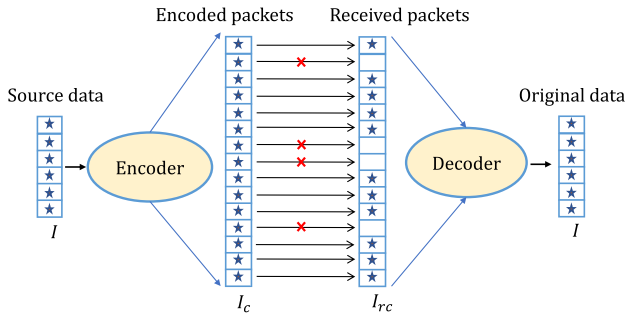


FIGURE 12. A pictorial illustration of the encoding/decoding process of rateless codes [104].

event of high packet loss [100]. To address these challenges, rateless codes (RCs) are designed to enable efficient transmission of data in erasure channels without relying on traditional explicit feedback mechanisms such as ACK/NACK. Instead, RCs utilize a different form of explicit feedback, known as a stop-bit, to indicate successful reception [15], [16]. This feedback message serves as an indicator to the sender that sufficient packets have been received for successful decoding. Rateless coding enables continuous transmission of encoded packets until the receiver sends a single feedback message confirming successful decoding of the entire message.

RCs in literature are often extensions of fountain codes for error-prone channel transmission [101]. Thus, rateless codes takes on the name fountain codes and other varieties [102]. Originally it was designed to provide efficient transmission of data in erasure channels and was limited primarily to multimedia applications specifically video streaming [101]. The rate of a code that is rateless can be viewed in two separate lights. i) instantaneous rate and ii) effective rate. Where the former is the ratio of the number of information expressed in bits to the total number of data sent over a particular time. While the latter is the rate of accumulated information at the instance of successful reception of the codeword [103].

The concept of rateless codes is depicted in Fig. 12. There, information data I from the source were encoded by the transmitter to generate a large size of encoded packets I_C which are then sent over an erasure channel. In the process, a number of packets get lost and only I_{rc} of the encoded packets arrives at the receiver. After the successful decoding, the original packet I is recovered.

Fountain codes have the advantage of encoding an infinite number of packets from n information symbols with low encoding and decoding complexity of the order $O(KlogK)$ [15], [16]. However, in multihop scenarios with packet loss, routing is not an optimal operation in fountain codes [105]. In such scenarios, network coding can be used at intermediate nodes to improve the performance. Batched sparse (BATS) codes are an example of a rateless sparse random linear network coding (RLNC) that incorporates network coding [105], [106].

Introducing network coding considerably enhances the robustness of rateless codes by increasing the dimension of

TABLE 2. Achieving URLLC despite Feedback loss: Approaches and Attributes.

Reference	Approach	Main Finding	Performance Metric
[68]	Introduced an information field consisting of a packet identifier (Pid), and Pid-$Echo$-$Reply$ to the network code acknowledgment (NC-ACK) header by which the sender can know the lost packet combunation(s)	Compared to standard TCP/NC [68], yields better performance but declines as the channel becomes bursty, retransmission decreased	Throughput
[99]	Optimized [69] by adding to the NC-ACK an 8-bit PLS that keeps records of ACKs from the oldest to the latest combination. So, every latest ACK carries the ACKs of previous combinations	If some feedback ACK is lost, the sender can the combination of information that was lost. Better performance than [69]. Sadly, the scheme is insufficient dense bursty channels	Goodput and decreased retransmission
[69]	Opposed to [100], authors proposed a 32-bit PLS, the cooperate use of ACK accumulator, ACK retransmission, and Duplicate generator to overcome dense burstiness	Compared to [69], there is increased bandwidth maximization, and performance is maintained despite ACK loss. This approach is more complex than the above schemes	goodput and bandwidth
[70]	A broadcast approach termed UTNC is used to service multiple receivers despite feedback loss based on a generation matrix (G). At each time slot an innovative packet is sent while satisfying weak-link receiver despite ACKs loss	Eradication of meaningless retransmissions caused by ACK loss. Decoding speed increased. Thus, throughput increased contrasting with previous scheme [101]	Throughput in terms of decoding speed and number of retransmissions

information served to the network. This leads to better error correction capabilities and more reliable communication networks. Thus, the combination of rateless codes and network coding provides an effective approach for transmitting data without the need for feedback [105], [106]. Refer to the following Section B-1 to catch a glimpse of [106] and [105].

B. THE INTERPLAY OF CODING STRATEGIES IN ACHIEVING URLLC WITH NO FEEDBACK

In blind coding where feedback is not required, FEC can be used to improve transmission success rate at the cost of packet redundancy. As depicted in Fig. 13, an increased number of redundant packets is necessary to compensate for lost packets, especially when the generation size grows larger. Without a feedback mechanism, the implementation of suitable coding design strategies becomes a difficult task. k-repetition strategy, sliding windows, multipaths, and flexible codes in connection with shortcodes can be implemented to achieve URLLC. We have generalized shortcodes in this survey to represent, short packet transmission, short block coding, and finite blocklength codes (i.e., short block length codes) which are all relevant to the delivery of URLLC. Thus, we have classified this heading based on the following; the interplay of sliding windows, shortcodes, long codes, and k-repetition strategy in URLLC, the interplay of sliding window and multi-path/multi-hop for URLLC, and the interplay of information size, overhead, field size, and payload delivery efficiency, fulcrum network coding: the interplay of GF(2) and GF(2^8) or GF(2^{16}) for satisfactory communication.

1) THE INTERPLAY OF SLIDING WINDOWS, SHORTCODES, LONG CODES, AND K-REPETITION STRATEGY IN URLLC

Due to the routing-based protocol limitation of rateless codes, which is not optimal, BATs codes have been proposed [105]. It merges RLNC into rateless codes (RC). However, with batched sparse (BATS) codes decoding cannot commence

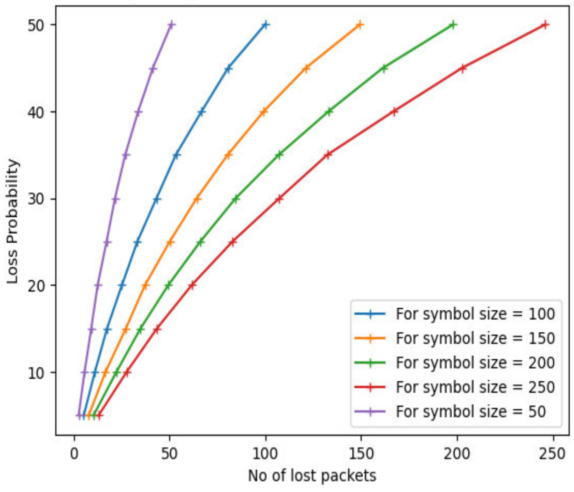


FIGURE 13. Effect of loss probability on the generation size.

until the receiver has complete full rank of coded packets causing long delays. This occurs especially in the case of Long codes (large information files). Overcoming this limitation requires that the n information size is broken into smaller blocks or short packets before encoding. But this also has the drawback of increasing the number of ACKs in ARQ-based protocols. In addition, due to the shorter length of short blocks, they are known to degrade decoding performance when adopted without a sliding window technique. Therefore, to achieve low latency and efficient decoding, the sliding window algorithm should be utilized in conjunction with short blocks

As a result, sliding-window BATs codes are proposed for low latency communication [106]. With this approach, decoded messages for each window can be delivered over to the application layer in an orderly manner. The virtual extension of the block created by the sliding window enables the code to achieve superior performance in terms of transmission delay, which can be valuable for applications where reducing delay, such as delay-sensitive video transmission, is crucial. Furthermore, compared to [105], the decoding performance of [106] increased due to the quality of distribution provided by the sliding window during transmission. In URLLC, short block coding is commonly used in conjunction with sliding window techniques to potentially achieve low latency and quick decoding. However, sliding window techniques can also be employed with other coding schemes and block lengths, depending on the specific system requirements.

Short block coding and the k-repetition strategy play a significant role in achieving reliable and low-latency communication. Transmitting a packet repeatedly improves the transmission success rate, reduces the number of retransmissions (in ARQ schemes), and lowers the decoding delay. These benefits come at the expense of bandwidth capacity [107], [108]. Coding over Short blocks is crucial to ultra-reliable and low-latency communication, as they are commonly used to transmit critical information with high reliability and low latency [109]. To improve the throughput of short-packet communication systems used in URLLC applications, Short block length codes have been employed [110], [111]. Short block length codes are significant for URLLC because they have a smaller block length, which reduces the decoding latency and complexity [112]. However, short block length codes have a higher error rate compared to long block length codes. Therefore, efficient decoders are required for short block length codes to achieve the desired error-rate performance and low decoding complexity [112]. Careful consideration of the block length is required to ensure that the coding rate is increased and the requirements of the communication system are met.

Reliability-latency tradeoffs have been investigated based on Short block length codes in 5G networks to deliver URLLC [113]. The Authors investigate the trade-offs between latency and reliability when the user equipment (UE) adopts i) transmission diversity (creating redundant packet replicas based on the number of resource blockss (RBs) in order to recover from link outage), and ii) varying set of RBs at transmission time intervalss (TTIs). Results show that the number of packet replicas created from transmission diversity −which is based on RBs allocation− affects the performance in terms of reliability and latency. Therefore, improved reliability is ensured as the number of replicated packets increases [114] at the expense of increased latency. In like manner, the reliability decreases as the RBs allocation increases but the latency is reduced. This is like the case of the k-repitition strategy which helps to create redundancy.

a: NETWORK CODING IMPROVES k-REPETITION STRATEGY AND USES SLIDING WINDOW TO ENHANCE DECODING PROBABILITY

The repetition strategy has been equally applied to boost the reliability of machine-type communication (MTC) systems desiring to transmit several short information packets using RLNC [114]. In comparison to the conventional k-repetition where each packet has to be transmitted k times in sequence, random linear network coding (RLNC) can at a single time slot transmit a linear combination of the coded packets equivalent to all the individual packets transmitted over transmission slots [114]. Results reveal that with RLNC, the k-repetition in the grant-free random access (GFRA) can decrease the needed time slot. Thus, reducing the access latency by 50% while meeting the desired reliability compared to the traditional k-repetition. However, its performance comes at the cost of complexity. Due to the added network coding overhead. Hence, obtaining a balance in the trade-off is still an ongoing research problem.

Furthermore, with respect to decoding error probability, network coding can improve communication reliability [115] at the expense of prolonged delay [105] compared to the famous k-repetition approach. Therefore, to overcome this limitation, sliding windows can help to provide the needed reliability-delay tradeoffs [115]. In this technique, linear combinations of raw data packets generated by sliding windows are sent alongside the original packets. This is similar to NC-ARQ-based schemes [20], [59], [60], [83]. A sliding NC scheme enables the decoder to recover a sequence of coded data packets on-the-fly [87], with an assigned particular decoding delay. Also, the decoding error rate was observed to decrease with increased k-repetition. However, compared to the conventional k-repetition, sliding network coding reduces the number of repetitions, and as a result, bandwidth is well utilized effectively. Thus, the gap in the performance of both schemes rises with an increase in k.

b: LONG CODES VS SHORTCODES: A TRADE-OFF BETWEEN LATENCY AND RELIABILITY

Having established the importance of shortcodes in achieving URLLC, a study has been carried out to predict the effect of encoding latency on varying generation and symbol size framing [87]. For clarity purposes, generation defines the number of original packets, while symbol size specifies the number of bytes in a piece of single data information. The authors used a counting argument to embark on the prediction. The investigation was tested on two scenarios of packet transmission such as On-the-fly (OF) and on full vector algorithms. However, in comparison with the full vector algorithm, the obtained results showed that the gain derived from using the OF algorithm is lower than expected since it involves fewer operations. Furthermore, depending on the alteration of either symbol size or generation size, one of them would greatly determine the observed changes in the encoding latency. This result is in agreement with [52] and [87]. Authors in rounding off suggested that this investigation be replicated across several (SIMD, CPU) platforms.

While adopting short block coding can help to meet certain time constraints and guarantee reliability in URLLC, it may not achieve full capacity. Thus, recently a systematic RS convolutional network coding with long code has been used to improve the transmission rate under equal limited time constraints while ensuring reliability [21]. The authors use systematic network coding [59], [60], [83] in order to reduce the delay and complexity. Each k data packet is divided into a group by the source node and using systematic RS convolutional code redundant check packets are generated. Then n coded packets are sent out. Convolutional codes have low decoding delay even though they are known for their long code property. Simulation results revealed that transmission rate reliability grows with increasing limited time constraints. It achieves almost the maximum delay of short block coding and the performance of long codes. However, the authors concluded that for small limited delay constraints, the use of short block coding is inevitable. We, therefore, conclude that to maintain optimal reliability and latency, the balance between applying short block coding and long codes must be determined. Table 3 gives a summary of some of the literature we reviewed on the relevance of sliding windows, shortcodes, long codes, and k-repetition strategy in achieving URLLC.

2) THE INTERPLAY OF SLIDING WINDOW AND MULTIPATH/MULTIHOP FOR URLLC

Sliding window network coding has been suggested in the accomplishment of URLLC using multipath communication [116]. Only two paths (LTE and WiFi) were considered by the authors. Based on a round-robin scheduling algorithm, systematic codes are sent over the network. From the packets contained in the sliding window, network-coded packets are transmitted right after sending uncoded packets through the paths. The authors maintained a fixed code rate for the entire transmission process. Simulation results showed that using both paths as against the best SP is far better when attempting to reduce latency compared to [93]. Also, an increase in the number of paths was reported to increase reliability just like in [65]. Furthermore, with increasing encoding window or generation size, errors experienced by previous packets can be recovered by coded packets at the cost of high delay which is consistent with [111]. Also, similar to [117] and [118], the authors stated that by applying the same code rate and window size, a lower delay is possible with sliding windows over finite blocks than it is with block code. Decreasing the code rate beyond the point at which losses close to zero are observed, shows a negligible influence on the delay. However, the scheme [116] achieved a better performance in terms of delay even at a higher code rate (lower redundancy) which translates into energy consumption reduction. Also, extending this study onto more than two paths is still in view. The prediction is that the performance of the scheme will be highly influenced by the most and least delay-prone paths as a result of delay asymmetry and the corresponding losses will average out over all paths. Future work in this direction would be to study the effect of redcoding at different paths using RLNC.

Due to the asymmetric nature of various paths in terms of how prone to delay they are, Packet reordering is still a prominent challenge that affects the effective utilization of bandwidth in multipath transmission and URLLC cannot be achieved without adequate channel utilization [66]. So to address this issue, skip NC (SNC) has been suggested [119]. SNC draws its concept from the interleaving strategy which group together and encode packets with a unique interleaving distance. The goal of SNC is to encode as many unique packets as possible rather than create as many unique encoded packets as possible. Simulation results prove that SNC provides decoding failure of a lower ratio, reduces the packet reordering issue, and makes for greater bandwidth utilization. All of these put together improve reliability. Furthermore, the number of retransmissions resulting from packet loss was reduced. The authors [119] also observed that at a 5% packet-loss rate, the overall throughput derived by SNC was enhanced by over 20%. Future consideration would be to develop a scheme that optimizes and analysis SNC and also there is the need to provide an algorithm for path-scheduling as well as multipath status monitoring.

a: ADDRESSING THE LIMITATIONS OF SLIDING WINDOW TO ACHIEVING URLLC IN A MULTIHOP SCENARIO

Having established that the sliding window-based random RLNC outperforms the block-based RLNC as it allows for immediate packet decodability and in addition, lowers the in-order delivery delay, it has only been applied to point-to-point networks. The reason is that the performance of sliding window-based schemes tends to degrade in multihop network settings due to recoding at intermediate nodes. Therefore, a scheme that supports the recoding of packets at the nodes has been presented, referred to as the sliding window NC (SWNC) based recoding algorithm [22]. The authors investigated the effects of infinite and finite coding window size on the SWNC performance. During recoding, early arrivals of (m first) packets or already decoded raw packets are forwarded to the next hop followed by c coded packets. Due to packet erasure, there are instances (idle time slots) where these intermediate nodes have no available new packet to forward. Three algorithms are provided to estimate the effect of idle time on the in-order delivery delay of packets, which includes i) the intermediate node does nothing within the idle time slot, ii) the second option is, to generate coded packets, and transmit rather than stay idle, and iii) the last alternative is to send partially decoded packets. If there are no packets available, the node transmits previously transmitted packets. Simulation results revealed that among these three algorithms, the least delay is associated with the second case where coded packets are sent within the idle time using a short finite window size. Also, in comparison with other schemes, SWNC recoding potential (in terms of per-packet delay) by 53% outclassed block-based RLNC. Hence, SWNC is fit for the constant streaming of packets with low latency. However, this SWNC was only simulated on a single-path multi-hop

TABLE 3. The interplay of sliding windows, shortcodes, long codes, and k-repetition strategy in URLLC.

Reference	Approach	Main Finding	Performance Metric
[105]	Proposed a BATs code that merges RLNC into rateless codes using long codes	Since it uses long codes, decoding cannot commence until the receiver has a complete full-rank of coded packet leading to long delays	Throughput and latency
[106]	Proposed a Sliding-window BATs code using short blocks for low latency communication is proposed	Yields a better performance than [107] in terms of average transmission delay and also allows in-order-delivery of packets	Transmission delay
[113]	Reliability-latency tradeoffs have been investigated based on Short block length codes in 5G networks to deliver URLLC	The number of packet replicas created from transmission diversity —which is based on RBs allocation— affects the performance in terms of reliability and latency.	Reliability and latency
[114]	Proposed the joint use of RLNC and k-repetition to boost the reliability of machine-type communication (MTC) systems transmitting several short information packets	Compared to the traditional k-repetition strategy, jointly using k-repetition and RLNC decreases the needed time slot, thus the access latency reduces by 50% while meeting the desired reliability. However, its performance comes at the cost of complexity	Reliability and latency
[115]	Proposed sliding network coding (SNC) to ensure high reliability without significantly increasing the decoding delay	Reduces excess replication of packets, as minimal replication makes for small transmission delay and effective bandwidth usage	Throughput in terms of decoding speed
[87]	Predicts the effect of encoding latency on varying generation and symbol size framing using a counting argument.	Symbol or generation size influences the encoding latency.	Latency
[21]	Proposed a systematic RS convolutional network coding with long codes to improve the transmission rate under limited time constraints	Convolutional codes have low decoding delay. It achieves almost the maximum delay of short block coding and the performance of long codes. However, the authors concluded that for small limited delay constraints, the use of short blocks is inevitable.	Reliability and latency

network scenario. Therefore, there is a need to determine the effect of recoding at the intermediate node in a multi-path multi-hop network.

Therefore, similar to [22], rather than frequently generating and transmitting already recoded packets, as suggested in [120], intermediate nodes are made to first send a subset of the just received coded/uncoded symbols and right after that, recoded symbols are generated from that same subset and transmitted. Thus, the recoding process employs different window sizes. With this approach, the receiver is able to immediately start the decoding process since the majority of the transmitted symbols from the source are uncoded. The outcome is low-latency communication while preserving the unique attributes of the sliding window. The result shows that, as against the recoding window scheme with a fixed size that gains by a factor of 3, this scheme further improves upon it by a factor of 2.

The effect of sliding window on multihop for boosting packet resilience to loss while guaranteeing low latency is presented in [121]. The multi-hops were represented by 6 fast-moving cars (indicative of a dynamically changing network) separated by a defined distance and speed of $100-300m$ and $100-150km/h$ respectively. The packet reception ratio (PRR) using IEEE 802.IP standard was used for the measurement of different rates. Then the measured PRR was used to simulate an actual channel scenario and the sliding window was applied. At the intermediate nodes, something slightly different was done, similar to [22]. Uncoded packets alongside partially coded (systematic) packets are sent after which the sliding window generates a RLNC code and transmits [121]. The simulation result shows that the degree of packet erasure can be kept below a maximum of 15% with a probability of 85% right up until the 4^{th} hop (vehicle). Furthermore, 75% of the resulting delay fell below the aggregate number of packets transmitted. Therefore, not only did the scheme proposed by the authors achieve reliability but low latency. However, the packet-delivery delay was observed to slightly increase towards the tail end of the transmission. Future work based on this would be to investigate the effect of dynamically changing the size of the coding window when using heterogeneous data rates for transmission.

The obvious limitation of sliding window in a multihop network has been addressed using two joint techniques, which include; caterpillar RLNC (CRLNC) and Full-vector RLNC (FvRLNC) [122]. Both schemes jointly enable intermediate nodes to recode while preserving the property of sliding windows and ensuring both reliability and latency. First, the message was encoded via CRLNC and transmitted by the source node. The intermediate nodes, upon receiving the encoded message, perform recoding based on two strategies: i) intermediate nodes get to send the first subsets of the oldest packet arrivals and afterward transmit decoded packets. Then coded packets of the sent packets are generated and sent to the next hop. This approach is similar to [22], and ii) the second employed algorithm implemented is the FvRLNC [123]. Based on the later algorithm, coded packets of all received packets — combined together — are generated and broadcast. Simulation results showed that aside from the source node, four other intermediate nodes must broadcast likewise so as to have the average packet loss reduced to around 4% when using the CRLNC in all the transmitting nodes (vehicles). Alternatively, when CRLNC is used jointly with FvRLNC for recoding, if 3 out of 6 intermediate nodes do the broadcast, the average packet loss drops by 0.2%. The authors also compared their result with the NoCode (store and forward) and reported packet loss of 20%, Thereby making the joint cooperation of CRLNC and FVRLNC suitable for increasing reliability without compromising latency in any way. Table 4 gives a summary of the literature we reviewed on the interplay of Sliding Window and Multipath/Multihop for URLLC.

3) THE INTERPLAY OF INFORMATION SIZE, OVERHEAD, AND FIELD SIZE FOR PAYLOAD DELIVERY EFFICIENCY

Selection of network code parameters has been found to have an undeniable impact on reliability, packet resilience to error, latency, or bandwidth, and the extent of coding complexity [54]. In a mesh (multihop-multipath) network where billions of interconnected devices are present, the need to accurately deliver information to the right recipient in good time comes at the price of overhead encapsulation. This overhead can be greater than the actual message targeted at the receiving node. For instance, to accurately deliver 25 bytes of real-time data, 48 additional bytes may have to be included to accommodate several accompanying protocol headers. Overhead resulting from RLNC due to the recoding at the intermediate nodes, is a function of the encoding vector size. So higher generation size implies more coefficients will be sent which declines the payload delivery efficiency. This added overhead it incurs, despite its great advantages, raises a concern. Field size also shows a great impact on the overhead size. Thus, decreasing the data information size that traverses through the network can result in a shortened network interface activity, particularly in a mesh topology. This, in turn, has a significant impact on reducing energy consumption [124]. Additionally, reducing the message size lowers the probability of bit error, which ultimately leads to lower latency.

Therefore, studies aimed at reducing the overhead in the network such as the joint consideration of RLNC and opportunistic routing (OpR) have been suggested [125]. According to recent research, it is possible to reduce OpR messages by at least 50% (and sometimes even more) using a customized compression technique based on robust header compression version 2 (RoHCv2) [126]. The integration of a network-coded header compression scheme that reduces the RLNC overhead and satisfies the URLLC has been undertaken in [127]. Basically, the compression process is hierarchical in the sense that, after the original stream is been compressed and resolved, it is then pushed into the RLNC algorithm for encoding. Both stages are independent of each other which gives it the benefit of flexibility in selecting coding parameters (generation size, systematic approach, field size, etc.) and coding schemes such as full-vector NC, sliding window, or online NC [128]. Results revealed that the size of the coded packet generated for transmission does not depend on the generation size, since the header of the coefficient that should have been transmitted is been replaced by one constant seed. This doubles the delivery efficiency of the payload and achieved a result of about 15% better in comparison to RoHCv2 [126]. According [129] reveal that at GF(4), the highest maximum effective rate and high completion time is reported among other field sizes. On the other hand, at GF(2^8), the least completion time is recorded and the amount of communicated information to destination users is small due to the large overhead. So, this means that the effect of the choice of field size on communication delay may come at the cost of overhead.

4) FULCRUM NETWORK CODING: THE INTERPLAY OF GF(2) AND GF(2^8) OR GF(2^{16}) FOR SATISFACTORY COMMUNICATION

According to [130] fulcrum NC (FNC) also demonstrates impressive flexibility just like [127] in its coding configuration that can achieve a great fit in Communication networks in terms of overhead reduction, to almost n bits given a generation size of $n + r$ packets and reduction of network complexity since relay nodes employ $GF(2)$ for recoding of packets. Fulcrum network coding aims to achieve the benefits of a high-field coding system using RLNC in a more efficient way. It offers high end-to-end (E2E) performance to high-capacity client devices while satisfying low-capacity devices. The flexibility inherent in FNC makes it relevant to a heterogeneous pool of client devices. Fulcrum Codes utilizes a systematic code. A version of the original data (of n packets) to be transmitted is encoded with a higher field size $GF(2^h)$ where $h > 1$. The outcome yields an expansion packet (r) or the outer code. This is then concatenated with the original data resulting in $n + r$ packets. The $n + r$ packets are now considered as a new generation of packets which are then recoded by the relay nodes with $GF(2)$ while traversing the network. At the destination, a receiver with higher capacity can with at least n number of received coded packets recover the original data using $GF(2^8)$ decoder, known as the outer decoder. Whereas, a receiver with low computing power engages the inner decoder ($GF(2)$ decoder) which requires a rank size of $(n + r)$ coded packets to reconstruct the original data. Decoding in $GF(2)$ though not complex (utilizes simple XOR operation) comes at a delay cost.

However, the analysis presented in [130] showed that the added dimension resulting from the expansion packet (r) helps regulates the trade-off between overhead, delay, and computational complexity. While recoding with $GF(2)$, larger size of r makes for a higher decoding probability because the risk of generating linearly dependent packets is reduced. Fulcrum network coding shows some advantages

TABLE 4. The interplay of sliding window and Multipath/Multihop for URLLC.

Reference	Approach	Main Finding	Performance Metric
[116]	Suggested sliding window network coding for accomplishing URLLC using multipath communication. Based on a round-robin fashion packets were transmitted via the paths	Using both paths as against the best single path is far better when attempting to reduce latency. An increase in the number of paths led to an increase in reliability. Better delay performance even at a higher code rate (lower redundancy)	Reliability and latency
[119]	Skip network coding is proposed to aid packet reordering for effective multipath channel utilization using interleaving strategy	Provides a low decoding failure rate, reduces the packet reordering issue and makes for greater utilization of bandwidth. At a 5% packet-loss-rate, the overall throughput derived was enhanced by over 20%	Reliability and latency
[22]	Proposed a sliding window algorithm to support the recoding of packets at the intermediate nodes in a single-path multihop. During recoding, early arrivals of (m first) packets or already decoded raw packets are forwarded to the next hop followed by c coded packets.	In comparison with other schemes, its recoding potential (in terms of per-packet delay) outperformed block-based RLNC by 53%. It is good for constant streaming of packets with low latency	Latency
[120]	Similar to [22], intermediate nodes are made to first send a subset of the just received coded/uncoded symbols and right after, recoded symbols are generated from the same subset and transmitted. Thus, the recoding process employs different window sizes	In contrast to recoding with a fixed window, this scheme further improves upon it by a factor of 2	Latency
[121]	Uncoded packets alongside partially coded (systematic) packets are sent after which the sliding window generates RLNC code and is transmits	The degree of packet erasure can be kept below a maximum of 15% with a probability of 85% right up until the 4^{th} hop. 75% of the resulting delay fell below the total number of packets transmitted	Reliability and latency
[122]	Proposed 2-joint use of Carterpillar RLNC (CRLNC) and Full-Vector RLNC (FvRLNC). Intermediate nodes send the first subsets of the oldest packet arrivals and afterward transmit decoded packets. Then, coded packets of the sent packets are generated and sent to the next hop. Finally, coded packets of all received packets are generated and transmitted	Intermediate nodes must broadcast so as to have the average packet loss reduced to around 4% when using CRLNC in all the transmitting nodes. Alternatively, jointly adopting CRLNC and FvRLNC for recoding, if 3 out of 6 nodes broadcast, the average packet loss drops by 0.2%.	Reliability and latency

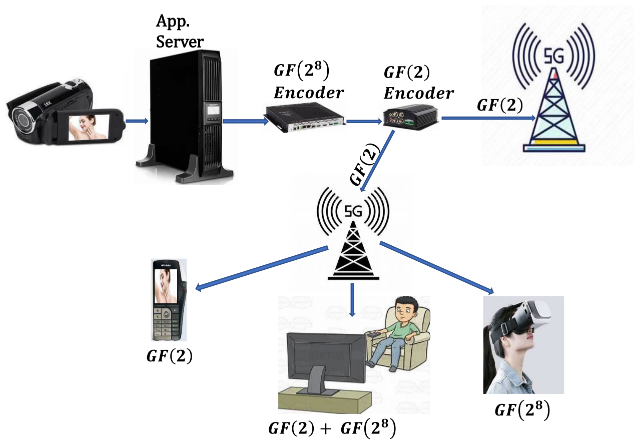


FIGURE 14. Fulcrum network codes enable high-performance operation at higher field sizes (i.e., GF(2^8)) while remaining compatible with GF(2)-only networks. Receivers have the option to balance decoding effort and delay by choosing to decode with GF(2) or in higher fields.

over other schemes. It requires low signaling overhead to convey the used coded coefficients to the receiver. Guarantees an increase in the coding processing speed by a factor of 20 [130] compared to the traditional RLNC [80].

As shown in Fig. 14, the special feature of fulcrum in terms of its ability to run on two different field sizes, enables the decoder to choose from either of $GF(2)$ or $GF(2^8)$ based on its computing capacity to decode the entire packets. However, utilizing a fixed field size for the entire decoding process irrespective of the received present status of the available packets [130] is suboptimal for end users, particularly in broadcasting situations where adopting a feedback mechanism is not feasible.

Hence, an adaptive algorithm that allows the decoder to dynamically switch between the two field sizes in response to the available number of received coded packets has been investigated [131]. This adaptive decoder hopes that its decision will lead to a successful recovery. Otherwise, it fails. The authors evaluated this scheme in a multicast environment where several receivers were targeted with information from a source. The unique property of this decoder shown by the simulation result is that it has a combination of low computation complexity and high decoding probability attributes contained in one decoder. The decoding probability showed an increase of approximately 40% compared to the inner decoders. Furthermore, the number of operations was reduced by approximately 30% compared to using only the outer decoder. furthermore, the decoding delay performance was better compared to [130]. Future work based on this approach is to inculcate ACKs to improve on the decoder's choice of field selection which may match up with the requirements of URLLC.

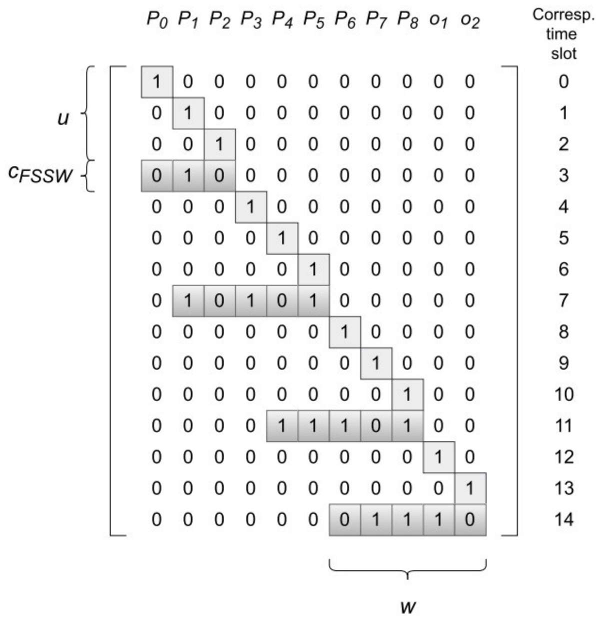
Extensive work on [131] has been carried out to provide an advance adaptive fulcrum-based decoder with the following contributions:

- they proposed a fulcrum decoder that adapts itself to the present channel state and user device computing strength [131].
- They proposed an advanced version of the fulcrum decoder to further improve the decoding probability while ensuring low decoding delay and overhead. Contrasting with earlier proposed fulcrum decoders, harnessing this scheme comes at the cost of a slight increase in complexity. Although in comparison to other RLNC and conventional FNC decoders, it showed less complexity.

The adaptive decoder in [131] had to wait for the generation of a packet to attain full rank before commencing the decoding process with respect to the receiver's choice of field size. This algorithm degrades the performance of the system in terms of delay. Another drawback is that the inner decoding could fail even if there are $n + r$ packets due to useless redundancy – an issue caused by packet linear dependency – which has been addressed by [132]. This is where the advanced version of the fulcrum decoder comes into play [37]. As soon as the inner decoding is declared unsuccessful, it has the ability to quickly switch onto the outer code (that requires only n coded packets and uses $GF(2^8)$) while in the midst of decoding. If there are sufficient n linearly independent packets, the decoding succeeds, hence improving the decoding probability and decoding delay at the expense of complexity. However, when the channel state is just fine, this advanced adaptive scheme accomplishes both low complexities and improved decoding probability. summarily, the decoding probability outclasses the singular use of the GF(2) decoder by around 60% while the number of operations is reduced by around 22% to 43% in comparison to RLNC GF(2^8) in a channel susceptible to error.

a: ADDRESSING FULCRUM LINEAR DEPENDENCY ISSUE

To solve the problem of linear dependency – which informs successful decoding – that plagues fulcrum network codes, Knowing that this can degrade its performance in terms of latency, a scheme that employs both dynamic sparsity and expansion packet (DSEP) to reduce the linear dependency issue of fulcrum codes has been proposed [132]. By varying the non-zero coding coefficients density and the added expansion packets (r), the coding complexity is pegged at a low while a high decoding probability is maintained. Simulation results showed that encoding/decoding throughput increased beyond 10-fold for the generation of large sizes in contrast to the normal fulcrum traditional approach. An increased fold from 1.7 – 4.3 in the decoding probability was likewise reported by the authors. The extension of DSEP to the explicit use of systematic code could be a great fit [46], [80], [133], [134].



$$\begin{array}{c|ccccccccccc|c}
 & P_0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & o_1 & o_2 & \text{Corresp. time slot} \\
\hline
u & 1&0&0&0&0&0&0&0&0&0&0 & 0 \\
u & 0&1&0&0&0&0&0&0&0&0&0 & 1 \\
u & 0&0&1&0&0&0&0&0&0&0&0 & 2 \\
c_{FSSW} & 0&1&0&0&0&0&0&0&0&0&0 & 3 \\
 & 0&0&0&1&0&0&0&0&0&0&0 & 4 \\
 & 0&0&0&0&1&0&0&0&0&0&0 & 5 \\
 & 0&0&0&0&0&1&0&0&0&0&0 & 6 \\
 & 0&1&0&1&0&1&0&0&0&0&0 & 7 \\
 & 0&0&0&0&0&0&1&0&0&0&0 & 8 \\
 & 0&0&0&0&0&0&0&1&0&0&0 & 9 \\
 & 0&0&0&0&0&0&0&0&1&0&0 & 10 \\
 & 0&0&0&0&1&1&1&0&1&0&0 & 11 \\
 & 0&0&0&0&0&0&0&0&0&1&0 & 12 \\
 & 0&0&0&0&0&0&0&0&0&0&1 & 13 \\
 & 0&0&0&0&0&0&0&1&1&1&0 & 14 \\
\hline
 & & & & & & & \multicolumn{5}{c}{w} & \\
\end{array}$$

FIGURE 15. Fulcrum systematic sliding window (FSSW) [135].

b: FULCRUM SLIDING WINDOW FOR LOW LATENCY COMMUNICATION

Recently, a sliding window-based fulcrum network coding scheme called Fulcrum Sliding Window (FSW) has been studied for low-latency communication [135]. This approach aims to address the large delays incurred by previously existing generation-driven RLNC fulcrum-based schemes during decoding and encoding processing. Two varieties of fulcrum sliding window (FSW) were proposed. The first uses non-systematic codes while the other employs systematic codes. Fig. 15 describes fulcrum systematic sliding window (FSSW). Given that u and C_{FSSW} denote a subset of uncoded packet and a coded packet respectively. While the outer coded packets ($r = 2$) are denoted by o_1 and o_2 and w defines the sliding window size. For the fulcrum systematic sliding window, after concatenating the generated outer coded packets, r to the N generation size packest, a subset of u (in this case, $u = 3$) uncoded packet is sent, accompanied by C_{FSSW} coded packets. This approach is similar to [20]. From Fig. 15, the coefficients corresponding to the uncoded packets are set to 1 while the rest in the row are set to 0. The coefficients of $C_{FSSW} = 1$ coded packets are randomly chosen from $GF(2)$ and transmitted. The window size consists of $w - u$ from the previously sent subset of uncoded packets since $w > u$. For example, the C_{FSSW} coded corresponding to time slot 7 comprises packets P_1 and P_2 from the first sent subset of uncoded packets. To further understand the decoding process and how the parameters are determined, look up pages 6 and 7 of [135]. Simply put, the

TABLE 5. Fulcrum network coding: the interplay of GF(2) and GF(2^8) or GF(2^{16}) for satisfactory communication.

Reference	Approach	Main Finding	Performance Metric
[130]	Proposed fulcrum network coding, that allows the server to satisfy end users based on their unique computational capacities	Using outer decoder in GF(2^8), high capacity users can with n out of $n + r$ coded packets recover the original packets while low capacity end-users will require at least $n + r$ coded packet to decode using inner decoder in GF(2)	Trade-off among overhead, delay, and computational complexity
[131]	Proposed an adaptive decoder that enables the decoder to adaptively alternate between GF(2) and GF(2^8) in response to the available coded packet status at the receiver	Yields low computation complexity and high decoding probability. Compared to the inner decoders, decoding probability rose by about 40%. Also, the number of operations was reduced by about 30% in contrast to engaging only the outer decoder. Better decoding delay performance than [132]	Decoding probability and delay
[37]	Advanced fulcrum decoder algorithm is proposed that is able to switch to the outer decoder immediately after the inner decoding fails due to linear dependency issues or erasure	Decoding probability and delay improve at the expense of complexity. Its decoding probability outclasses the singular use of the GF(2) decoder by around 60%. Also, the number of operations is reduced by around 22% to 43% in comparison to using just RLNC GF(2^8)	Decoding probability and delay
[132]	Proposed a scheme that employs both dynamic sparsity and expansion packet (r) to reduce the linear dependency issue of fulcrum codes	Encoding/decoding throughput increased beyond 10 fold for generation of large sizes compared in contrast to Fulcrum RLNC. Authors reported an increased fold from 1.7–4.3 in the decoding probability	Decoding probability
[135]	Since generation-based fulcrum codes incurs large delays, Sliding window fulcrum is proposed	Fulcrum systematic sliding window greatly reduced the in-order delivery delay of packets (particularly for moderately large generation and window sizes less than $(\frac{1}{4})^{\text{th}}$). Using systematic codes helps to reduce linear dependency issue	In-order delivery delay

decoder will only require the coded packet to recover from any erasure since uncoded packets are included in the transmission. In contrast with the non-systematic SW approach, simulation results revealed that Fulcrum systematic sliding window greatly reduces the in-order delivery delay of packets. However, that is only true for moderately large generation and window sizes less than $(\frac{1}{4})^{\text{th}}$. Similarly, the encoding and decoding throughput increased significantly compared to generation-based fulcrum network codes. We observe that the incorporation of uncoded packets into the transmission protocol will alleviate the linear dependency issue though not completely. Moreover, We found that the approach deployed in FSSW is deterministic. Therefore, it would be beneficial to consider incorporating ARQ into FSSW. Table 5 gives a summary of the literature we reviewed on the interplay of GF(2) and GF(2^8) or GF(2^{16}) for satisfactory communication.

V. DISCUSSION AND FUTURE PERSPECTIVES

We discuss in this section the summary and future perspectives of the surveyed studies on how network coding can help to realize URLLC despite feedback limitations for 5G and beyond.

A. ACHIEVING URLLC DESPITE DELAYED FEEDBACK

From the studies, Feedback delay in communication systems can be addressed using two approaches: causal and adaptive-causal. In Table 6, we connect the strengths of each approach to the specific requirements and challenges of IIoT:

1) CAUSAL APPROACH BASED ON PRESCRIBED CODE RATE

In IIoT, safety-critical applications like real-time monitoring of equipment or process control require low latency and high reliability. The causal approach based on systematic codes is well-suited for such applications, ensuring the timely delivery of crucial information and maintaining the desired level of reliability. However, contrary to the adaptive causal schemes, this approach is suboptimal when the channel becomes bursty because it relies on a deterministic code.

2) ADAPTIVE CAUSAL APPROACH BASED ON CHANNEL ESTIMATION

IIoT environments can be highly dynamic, with changing network conditions and varying levels of interference. The adaptive causal approach based on channel estimation can adapt to these changes, optimizing throughput according to the current network state while ensuring that communication remains reliable and has low latency. This adaptability makes it an attractive choice for applications such as predictive maintenance and remote monitoring in industrial settings. The complexity may increase if the algorithm inaccurately predicts channel state based on delayed feedback, leading to additional computational demands for refining predictions.

3) ADAPTIVE CAUSAL APPROACH BASED ON DEEP-LEARNING

IIoT systems often generate massive amounts of data, which can be leveraged to improve communication performance. The adaptive causal approach based on deep learning utilizes

TABLE 6. Comparison of network coding schemes for achieving URLLC despite delayed feedback.

Approach	Latency	Reliability	Throughput	Complexity	Withstand Burst Error	IIoT Benefits
Causal approach based on prescribed code rate	Low	High	High	Low	Low	Real-time monitoring, process control
Adaptive causal approach based on channel estimation	Low	High	High	Moderate	High	Predictive maintenance, remote monitoring
Adaptive Causal approach based on deep-learning	Low	High	High	High	High	High-performance IIoT systems
Adaptive causal approach based on queuing theory	Moderate	High	Moderate	Moderate	Moderate	Large-scale IIoT systems (smart factories, smart grid networks)
Multipath approach for achieving URLLC	Low	High	High	Moderate to High	High	Autonomous robots, intelligent transportation systems

this data to make better predictions about channel states, resulting in substantial gains in latency, reliability, and throughput. While the complexity of this approach may be a concern for some IIoT devices, it could be an ideal solution for high-performance IIoT systems where the benefits outweigh the increased complexity.

4) ADAPTIVE CAUSAL APPROACH BASED ON QUEUING THEORY

IIoT systems often involve the exchange of data between multiple devices, leading to the formation of queues. The adaptive causal approach based on queuing theory can effectively manage these queues, reducing latency and improving reliability. This approach is particularly well-suited for large-scale IIoT systems, such as smart factories or smart grid networks, where efficient queue management is crucial for maintaining optimal performance.

5) MULTIPATH APPROACH FOR ACHIEVING URLLC

In IIoT environments, network connections can be diverse, and leveraging multiple paths can provide significant performance improvements. The multipath approach for achieving URLLC can enhance communication performance by exploiting these multiple network paths. This approach is especially beneficial for IIoT applications that require high reliability and low latency, such as coordinating autonomous robots in a manufacturing facility or managing traffic in intelligent transportation systems.

In summary, the choice of the appropriate URLLC approach for IIoT depends on the specific requirements of the application, including latency, reliability, throughput, and complexity. By selecting the most suitable approach, IIoT systems can achieve the desired level of communication performance, enabling more efficient and effective industrial operations. A standout application in the literature we considered is the use of systematic codes to compensate for packet erasure and enable fast packet decoding, crucial for maintaining high reliability and low latency in IIoT networks.

B. ACHIEVING URLLC DESPITE FEEDBACK LOSS

Achieving ultra-reliable low-latency communication (URLLC) in the presence of feedback loss is a significant challenge, particularly in the context of the Industrial Internet of Things (IIoT). Ensuring high reliability and low latency between IIoT devices, sensors, and actuators is crucial for supporting mission-critical applications and time-sensitive processes. In order to address this challenge, various communication schemes have been proposed, each with its own advantages, limitations, and complexities.

Considering Table 7, TCP/NCwBLT and UTNC emerge as the most promising approaches for IIoT applications due to their ability to handle feedback loss effectively, despite their high complexity. Although in the literature, the complexity of UTNC appears to be low, as only two receivers were used to simulate the results. However, we predict that as the number of receivers increases, the complexity may also increase, and a receiver with the poorest connection can cause performance to decline. The benefits these methods provide in terms of goodput and decoding latency can lead to more reliable and responsive communication links in IIoT systems. This, in turn, can result in improved operational efficiency, better monitoring and control of industrial processes, and enhanced safety in industrial environments.

In our review process, we found that not many studies have considered channels with bidirectional loss. Hence, there is a need to unravel more strategies that could be helpful in achieving low-latency communication even in the presence of feedback loss.

C. THE INTERPLAY OF CODING STRATEGIES IN ACHIEVING URLLC WITH NO FEEDBACK

A blind network coding-based scheme bypasses the RTT delay since it functions without any feedback mechanism. Therefore, to achieve URLLC, Table 8 gives an overall view of the interplay of various parameters which we explain as follows.

1) THE INTERPLAY OF SLIDING WINDOWS, SHORTCODES, LONG CODES, AND K-REPETITION STRATEGY IN URLLC

This approach focuses on dynamically adjusting the window size, adaptive coding rate, packet replication, and rateless coding. The main advantages are improved reliability, low latency, and efficient resource allocation. However, there are challenges in selecting the optimal code and window size.

TABLE 7. Comparison of network Coding schemes for achieving URLLC despite feedback loss.

Metric	TCP/NC	TCP/NCwLRLBE	TCP/NCwBLT	UTNC
Goodput	Decreases with packet loss rate	Decreases with packet loss rate	Maintains irrespective of Ack loss rate	Increases by minimizing unnecessary retransmissions
Decoding Latency	Moderate	N/A	N/A	Reduces
Retransmission Rate	High with packet loss	N/A	Reduces wasteful retransmissions	Minimizes by utilizing 'drop-when-perceived' algorithm
Channel Utilization	Moderate	N/A	N/A	Improves
Robustness to Feedback Loss	Limited	Struggles with bursty feedback loss	Tolerates bidirectional loss	Maintains performance with feedback erasures
Complexity	Moderate	Less Complex	More Complex	Simpler approach

TABLE 8. The interplay of coding strategies in achieving URLLC with no feedback.

Interplay	Key Concepts	Advantages	Limitations	Future Directions	IIoT Benefits
Sliding Windows, Shortcodes, Long Codes, and K-Repetition Strategy	Dynamic window size adjustment, adaptive coding rate, packet replication, and rateless coding	Improved reliability, low latency, efficient resource allocation	Code selection and optimal window size	Balancing transmission rate and low latency	Real-time monitoring and control, efficient use of resources
Sliding Window and Multipath/Multihop for URLLC	Path diversity, adaptive path selection, link quality estimation, load balancing, joint routing, and coding	Enhanced throughput and reliability, better link utilization	Reordering issues, complexity	Improved coding schemes, reduced overhead	Improved connectivity and reliability, optimized link utilization
Information Size, Overhead, and Field Size for Payload Delivery Efficiency	Adaptive information size, dynamic overhead control, and field size optimization	Energy-efficient communication, lower latency	Increased overhead with higher generation size	Adaptive network code parameter selection	Energy-efficient communication in IIoT networks, lower latency for time-sensitive applications
Fulcrum Network Coding: GF(2) and(or) GF($2^{8 \text{ or } 16}$) for Satisfactory Communication	Heterogeneous client satisfaction, adaptive decoding, and sliding window	Flexibility, low complexity, high decoding probability, low overhead, and low latency using FSSW	Linear dependency issue, complexity in some scenarios, increased delay in GF(2)	Addressing linear dependency (although has been partly solved by DSEP and FSSW)	Supporting heterogeneous IIoT devices with different computational capabilities, flexible communication schemes, and improved latency with FSSW

Shortcodes, are crucial when facing small delay constraints. Although RS convolutional network codes can offer low latency for strict delay constraints, their performance may decline under certain conditions. Utilizing shortcodes helps meet small delay constraint requirements, while combining shortcodes with k-repetition further enhances reliability despite increased complexity due to overhead. An open challenge remains in closing the gap between capacity in terms of transmission rate and latency using shortcodes. Future directions involve developing an adaptive algorithm that alternates between long and short network codes depending on delay constraints, allowing for capacity achievement, delay reduction, overhead minimization, and complexity reduction. This will ensure efficient use of resources in IIoT networks and facilitate real-time monitoring and control.

2) THE INTERPLAY OF SLIDING WINDOW AND MULTIPATH/MULTIHOP FOR URLLC

Sliding windows and k-repetition have shown potential in achieving low delay and reliable communication in multipath and multihop settings for URLLC. The strategy emphasizes path diversity, adaptive path selection, link quality estimation, load balancing, joint routing, and coding, which leads to enhanced throughput, reliability, low latency, and better link utilization. However, challenges still remain, such as recoding issues at intermediate nodes in a multi-hop scenario, packet reordering, and the effects of delay asymmetry in multipath. The main limitations are reordering issues and increased complexity. The optimal approach for forwarding received packets in a multipath-multihop network has not been fully explored, leaving room for future research. Furthermore, while ARQ schemes can adapt and balance transmission rates based on feedback, blind coding lacks this capability. Investigating novel strategies like interleaving and round-robin packet scheduling can help address the reordering issue. Additionally, future work should study the effect of recoding at different paths using RLNC to better understand the performance implications of delay asymmetry and the resulting losses across all paths. Improved coding schemes and reduced overhead are future directions to improve connectivity and reliability while optimizing link utilization in IIoT networks.

3) THE INTERPLAY OF INFORMATION SIZE, OVERHEAD, AND FIELD SIZE FOR PAYLOAD DELIVERY EFFICIENCY

Achieving URLLC is attainable but comes at a price overhead, computational complexity, increased energy consumption, or increased channel resources. Therefore this interplay presents how information size, dynamic overhead control, and field size optimization can influence payload delivery efficiency. It leads to energy-efficient communication and lower latency. The main limitation is increased overhead with higher generation size. Future directions include adaptive network code parameter selection to ensure energy-efficient communication in IIoT networks, as well as reduced latency for time-sensitive applications. Thus, the next paragraph discusses fulcrum network code which has the unique property of adapting its field size to the network status.

4) FULCRUM NETWORK CODING: THE INTERPLAY OF GF(2) AND GF(2^8) OR GF(2^8) FOR SATISFACTORY COMMUNICATION

This strategy focuses on heterogeneous client satisfaction, adaptive decoding, and sliding window techniques. It provides flexibility, low complexity, high decoding probability, low overhead, and low latency, especially using FSSW. However, there are challenges related to complexity in some scenarios, and increased delay when decoding in GF(2). Furthermore, the use of GF(2) results in overhead reduction at the cost of increased linear dependency [129]. Although dynamic sparsity and expansion packet DSEP have been proposed to address this challenge it is still not completely solved. However, due to the recently proposed fulcrum systematic sliding window sliding (FSSW), the linear dependency effect is reduced. This is because uncoded packets are sent followed by coded packets. So, the decoder will only require the GF(2) coded packets when an uncoded packet is lost. Thus, this approach guarantees low-latency communication. To further close this gap (optimizing the linear dependency issue), it will be a great idea to implement a fulcrum based on (i) ARQ scheme to further make up for the linear dependency challenge (ii) instead of GF(2), engage GF(2^2) using composite fields to allow for the use of $0's$ and $1's$ as coefficients.

VI. CONCLUSION

In this paper, we demonstrate how certain network coding schemes can provide URLLC benefits, even with feedback delay, loss, or absence. This is particularly relevant for mission-critical systems like IIoT. We categorize these schemes based on the two primary approaches (TCP and UDP) used for packet protection while ensuring URLLC. Under the TCP schemes, we categorized them into two, which includes i) Achieving URLLC despite feedback delay, and ii) Achieving URLLC despite delayed feedback loss. For the UDP, we grouped them into four which comprise i) The interplay of sliding windows, shortcodes, long codes, and k-repetition strategy in URLLC, ii) The interplay of Sliding Window and Multipath/Multihop for URLLC, iii) The interplay of information size, overhead, and field size for payload delivery efficiency, and iv) Fulcrum network coding: The interplay of GF(2) and GF($2^{8 \text{ or } 16}$) for satisfactory communication. Based on our survey, network coding alone may not be able to achieve URLLC in the presence of feedback limitations. However, it has been shown that combining network coding with other techniques such as systematic codes, sliding windows, shortcodes, etc., can improve the performance of URLLC. It is important to note that achieving the desired URLLC specification often involves tradeoffs between factors such as latency, reliability, energy efficiency, and overhead, and therefore requires careful consideration and optimization. Therefore, the prevalent issues inherent in the application of network coding can be significantly alleviated only to an extent but not totally.

REFERENCES

[1] M. Zverev, R. Aguero, P. Garrido, and J. Bilbao, “Network coding for IIoT multi-cloud environments,” in *Proc. 9th Int. Conf. Internet Things*, Oct. 2019, pp. 1–4.

[2] S. Mumtaz, A. Bo, A. Al-Dulaimi, and K.-F. Tsang, “Guest editorial 5G and beyond mobile technologies and applications for industrial IoT (IIoT),” *IEEE Trans. Ind. Informat.*, vol. 14, no. 6, pp. 2588–2591, Jun. 2018.

[3] M. Matin, *Network Coding*. London, U.K.: IntechOpen, 2018.

[4] S. R. Pokhrel, J. Jin, and H. L. Vu, “Mobility-aware multipath communication for unmanned aerial surveillance systems,” *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 6088–6098, Jun. 2019.

[5] S. R. Pokhrel, N. Kumar, and A. Walid, “Towards ultra reliable low latency multipath TCP for connected autonomous vehicles,” *IEEE Trans. Veh. Technol.*, vol. 70, no. 8, pp. 8175–8185, Aug. 2021.

[6] P. Popovski, J. J. Nielsen, C. Stefanovic, E. D. Carvalho, E. Strom, K. F. Trillingsgaard, A.-S. Bana, D. M. Kim, R. Kotaba, J. Park, and R. B. Sorensen, “Wireless access for ultra-reliable low-latency communication: Principles and building blocks,” *IEEE Netw.*, vol. 32, no. 2, pp. 16–23, Mar. 2018.

[7] G. P. Fettweis, “The tactile internet: Applications and challenges,” *IEEE Veh. Technol. Mag.*, vol. 9, no. 1, pp. 64–70, Mar. 2014.

[8] *Study on Enhancement of URLLC Supporting in 5GC*, document TS 23.725, 3GPP, Mar. 2018.

[9] *Service Requirements for the 5G System*, document TS 22.26, 3GPP, 2019.

[10] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, “What will 5G be?” *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, Jun. 2014.

[11] J. K. Sundararajan, D. Shah, and M. Medard, “ARQ for network coding,” in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2008, pp. 1651–1655.

[12] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, “XORs in the air: Practical wireless network coding,” in *Proc. Conf. Appl., Technol., Architectures, Protocols Comput. Commun.*, 2006, pp. 243–254.

[13] L. Keller, E. Drinea, and C. Fragouli, “Online broadcasting with network coding,” in *Proc. 4th Workshop Netw. Coding, Theory Appl.*, Jan. 2008, pp. 1–6.

[14] Y. Cho, S. Karande, K. Misra, H. Radha, J. Yoo, and J. Hong, “On channel capacity estimation and prediction for rate-adaptive wireless video,” *IEEE Trans. Multimedia*, vol. 10, no. 7, pp. 1419–1426, Nov. 2008.

[15] M. Luby, “LT codes,” in *Proc. 43rd Annu. IEEE Symp. Found. Comput. Sci.*, Nov. 2002, p. 271.

[16] A. Shokrollahi, “Raptor codes,” *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.

[17] C. Fragouli, D. Lun, M. Medard, and P. Pakzad, “On feedback for network coding,” in *Proc. 41st Annu. Conf. Inf. Sci. Syst.*, Mar. 2007, pp. 248–252.

[18] P. Pakzad, C. Fragouli, and A. Shokrollahi, “Coding schemes for line networks,” in *Proc. Int. Symp. Inf. Theory*, 2005, pp. 1853–1857.

[19] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, “Network information flow,” *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[20] J. Cloud, D. Leith, and M. Médard, "A coded generalization of selective repeat ARQ," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 2155–2163.

[21] W. Guo, W. Zhang, W. Zhang, and B. Bai, "Convolutional network coding based on systematic RS code for low-latency guarantee," in *Proc. Comput., Commun. IoT Appl.*, Nov. 2021, pp. 232–237.

[22] 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.

[23] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5G: RAN, core network and caching solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3098–3130, 4th Quart., 2018.

[24] D. Rico and P. Merino, "A survey of end-to-end solutions for reliable low-latency communications in 5G networks," *IEEE Access*, vol. 8, pp. 192808–192834, 2020.

[25] I. Al-Anbagi, M. Erol-Kantarci, and H. T. Mouftah, "A survey on cross-layer quality-of-service approaches in WSNs for delay and reliability-aware applications," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 525–552, 1st Quart., 2016.

[26] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. Allerton Conf. Commun., Control, Comput.*, vol. 41, 2003, pp. 40–49.

[27] T. Ho and D. Lun, *Network Coding: An Introduction*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[28] P. Chou and Y. Wu, "Network coding for the internet and wireless networks," *IEEE Signal Process. Mag.*, vol. 24, no. 5, pp. 77–85, Sep. 2007.

[29] Y.-J. Chen, L.-C. Wang, K. Wang, and W.-L. Ho, "Topology-aware network coding for wireless multicast," *IEEE Syst. J.*, vol. 12, no. 4, pp. 3683–3692, Dec. 2018.

[30] D. Jiang, Z. Xu, W. Li, and Z. Chen, "Network coding-based energy-efficient multicast routing algorithm for multi-hop wireless networks," *J. Syst. Softw.*, vol. 104, pp. 152–165, Jun. 2015.

[31] T. Etzion and A. Wachter-Zeh, "Vector network coding based on subspace codes outperforms scalar linear network coding," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 2460–2473, Apr. 2018.

[32] Q. T. Sun, X. Yang, K. Long, X. Yin, and Z. Li, "On vector linear solvability of multicast networks," *IEEE Trans. Commun.*, vol. 64, no. 12, pp. 5096–5107, Dec. 2016.

[33] J. Ebrahimi and C. Fragouli, "Algebraic algorithms for vector network coding," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 996–1007, Feb. 2011.

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

[35] A. Douik, S. Sorour, T. Y. Al-Naffouri, H.-C. Yang, and M.-S. Alouini, "Delay reduction in multi-hop device-to-device communication using network coding," *IEEE Trans. Wireless Commun.*, vol. 17, no. 10, pp. 7040–7053, Oct. 2018.

[36] 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.

[37] V. Nguyen, J. A. Cabrera, D. You, H. Salah, G. T. Nguyen, and F. H. P. Fitzek, "Advanced adaptive decoder using fulcrum network codes," *IEEE Access*, vol. 7, pp. 141648–141661, 2019.

[38] R. Naumann, M. Schaeffer, S. Dietzel, and B. Scheuermann, "Prioritize efficiently: Layer selection in network coding," *Comput. Commun.*, vol. 144, pp. 100–111, Aug. 2019.

[39] N. I. Sulieman, E. Balevi, K. Davaslioglu, and R. D. Gitlin, "Diversity and network coded 5G fronthaul wireless networks for ultra reliable and low latency communications," in *Proc. IEEE 28th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Oct. 2017, pp. 1–6.

[40] N. Siasi, N. I. Sulieman, and R. D. Gitlin, "Ultra-reliable NFV-based 5G networks using diversity and network coding," in *Proc. IEEE 19th Wireless Microw. Technol. Conf. (WAMICON)*, Apr. 2018, pp. 1–4.

[41] S. Emara, S. L. Fong, B. Li, A. Khisti, W.-T. Tan, X. Zhu, and J. Apostolopoulos, "Low-latency network-adaptive error control for interactive streaming," *IEEE Trans. Multimedia*, vol. 24, pp. 1691–1706, 2022.

[42] I. E. Qachchach, O. Habachi, J.-P. Cances, and V. Meghdadi, "Efficient multi-source network coding using low rank parity check code," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2018, pp. 1–6.

[43] 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.

[44] H. Shojania and B. Li, "Parallelized progressive network coding with hardware acceleration," in *Proc. 15th IEEE Int. Workshop Quality Service*, Jun. 2007, pp. 47–55.

[45] X. Chu, K. Zhao, and M. Wang, "Massively parallel network coding on GPUs," in *Proc. IEEE Int. Perform., Comput. Commun. Conf.*, Dec. 2008, pp. 144–151.

[46] J. Heide, M. V. Pedersen, F. H. P. Fitzek, and T. Larsen, "Network coding for mobile devices—Systematic binary random rateless codes," in *Proc. IEEE Int. Conf. Commun. Workshops*, Jun. 2009, pp. 1–6.

[47] H. Shojania, B. Li, and X. Wang, "Nuclei: GPU-accelerated many-core network coding," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 459–467.

[48] M. Jafari, L. Keller, C. Fragouli, and K. Argyraki, "Compressed network coding vectors," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2009, pp. 109–113.

[49] D. Gligoroski, K. Kralevska, and H. Øverby, "Minimal header overhead for random linear network coding," in *Proc. IEEE Int. Conf. Commun. Workshop (ICCW)*, Jun. 2015, pp. 680–685.

[50] J. Heide and D. Lucani, "Composite extension finite fields for low overhead network coding: Telescopic codes," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 4505–4510.

[51] J. Heide and D. E. Lucani, "Composite extension finite fields for low overhead network coding," U.S. Patent 9 787 614, Oct. 10, 2017.

[52] R. Torre, C. C. Sala, S. Pandi, H. Salah, G. T. Nguyen, and F. H. P. Fitzek, "A random linear network coded HARQ solution for lossy and high-jitter wireless networks," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Dec. 2020, pp. 1–6.

[53] X. Zhao, "Notes on 'exact decoding probability under random linear network coding,"' *IEEE Commun. Lett.*, vol. 16, no. 5, pp. 720–721, May 2012.

[54] V. Latzko, C. Vielhaus, and F. H. P. Fitzek, "Usecase driven evolution of network coding parameters enabling tactile internet applications," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.

[55] M. Toemoeskoezi, F. H. P. Fitzek, D. E. Lucani, M. V. Pedersen, and P. Seeling, "On the delay characteristics for point-to-point links using random linear network coding with on-the-fly coding capabilities," in *Proc. 20th Eur. Wireless Conf.*, May 2014, pp. 1–6.

[56] B. Sklar, *Digital Communications*, vol. 2. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.

[57] J. K. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher, and J. Barros, "Network coding meets TCP," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 280–288.

[58] P. Elias, "Coding for two noisy channels," in *Proc. 3rd London Symp. Inf. Theory*, Sep. 1955, pp. 61–76.

[59] A. Cohen, D. Malak, V. B. Bracha, and M. Médard, "Adaptive causal network coding with feedback," *IEEE Trans. Commun.*, vol. 68, no. 7, pp. 4325–4341, Jul. 2020.

[60] A. Cohen, M. Médard, and S. Shamai (Shitz), "Broadcast approach meets network coding for data streaming," 2022, *arXiv:2202.03018*.

[61] A. Schneuwly, D. Malak, and M. Médard, "Discrete water filling multipath packet scheduling," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 1658–1663.

[62] 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.

[63] P. Garrido, D. J. Leith, and R. Aguero, "Joint scheduling and coding for low in-order delivery delay over lossy paths with delayed feedback," *IEEE/ACM Trans. Netw.*, vol. 27, no. 5, pp. 1987–2000, Oct. 2019.

[64] R. G. L. D'Oliveira, A. Cohen, J. Robinson, T. Stahlbuhk, and M. Médard, "Post-quantum security for ultra-reliable low-latency heterogeneous networks," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Nov. 2021, pp. 933–938.

[65] S. Pokhrel, N. Kumar, and A. Walid, "Towards ultra reliable low latency multipath TCP for connected autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 70, no. 8, pp. 8175–8185, Aug. 2021.

[66] Q. Zhang and X. Xu, "Low-latency FEC design with geometric packet arrival and delayed feedback," *IEEE Commun. Lett.*, vol. 25, no. 4, pp. 1139–1143, Apr. 2021.

[67] J. K. Sundararajan, D. Shah, M. Médard, 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.

[68] N. Viet Ha, K. Kumazoe, and M. Tsuru, "TCP network coding with adapting parameters for bursty and time-varying loss," *IEICE Trans. Commun.*, vol. 101, no. 2, pp. 476–488, 2018.

[69] N. V. Ha, T. T. T. Nguyen, and M. Tsuru, "TCP with network coding enhanced in bi-directional loss tolerance," *IEEE Commun. Lett.*, vol. 24, no. 3, pp. 520–524, Mar. 2020.

[70] Y. Gao, N. Zhang, and G. Kang, "A novel online network coding scheme for broadcast channels with imperfect feedback and decoding delay constraints," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Aug. 2018, pp. 448–453.

[71] B. Shang, L. Liu, H. Song, B. Xu, S. Pudlewski, and E. S. Bentley, "Trade-offs in reliability, latency, and energy for random network coding-enabled networks," *IEEE Commun. Lett.*, vol. 25, no. 8, pp. 2768–2772, Aug. 2021.

[72] 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.

[73] F. Gabriel, S. Wunderlich, S. Pandi, F. H. P. Fitzek, and M. Reisslein, "Caterpillar RLNC with feedback (CRLNC-FB): Reducing delay in selective repeat ARQ through coding," *IEEE Access*, vol. 6, pp. 44787–44802, 2018.

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

[75] X. Xu, Y. Zeng, Y. Li, and B. Vucetic, "Minimum-latency FEC design with delayed feedback: Mathematical modeling and efficient algorithms," *IEEE Trans. Wireless Commun.*, vol. 19, no. 11, pp. 7210–7223, Nov. 2020.

[76] C. Peng Fu and S. C. Liew, "TCP Veno: TCP enhancement for transmission over wireless access networks," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 2, pp. 216–228, Feb. 2003.

[77] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP westwood: Bandwidth estimation for enhanced transport over wireless links," in *Proc. 7th Annu. Int. Conf. Mobile Comput. Netw.*, Jul. 2001, pp. 287–297.

[78] *Study on New Radio Access Technology Physical Layer Aspects*, document 38.802, V14. 2.0, 3GPP, 2017.

[79] M. Kwon and H. Park, "Analysis on decoding error rate of systematic network coding," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2017, pp. 258–259.

[80] S. Pandi, F. Gabriel, J. A. Cabrera, S. Wunderlich, M. Reisslein, and F. H. P. Fitzek, "PACE: Redundancy engineering in RLNC for low-latency communication," *IEEE Access*, vol. 5, pp. 20477–20493, 2017.

[81] S. Liu, Z. Chen, J. Xie, L. Liang, M. Wang, and Y. Jia, "Optimal power allocation of cooperative NOMA with finite blocklength codes," in *Proc. Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2020, pp. 1200–1205.

[82] S. S. Ullah, S. C. Liew, G. Liva, and T. Wang, "Implementation of short-packet physical-layer network coding," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 284–298, Jan. 2023.

[83] A. Cohen, A. Solomon, and N. Shlezinger, "DeepNP: Deep learning-based noise prediction for ultra-reliable low-latency communications," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2022, pp. 2690–2695.

[84] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley, "How hard can it be? Designing and implementing a deployable multipath TCP," in *Proc. 9th USENIX Symp. Networked Syst. Design Implement.*, 2012, pp. 399–412.

[85] C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure, "Experimental evaluation of multipath TCP schedulers," in *Proc. ACM SIGCOMM Workshop Capacity Sharing Workshop*, Aug. 2014, pp. 27–32.

[86] Y.-C. Chen and D. Towsley, "On bufferbloat and delay analysis of multipath TCP in wireless networks," in *Proc. IFIP Netw. Conf.*, Jun. 2014, pp. 1–9.

[87] L. Nielsen, R. R. Hansen, and D. E. Lucani, "Latency performance of encoding with random linear network coding," in *Proc. 24th Eur. Wireless Conf.*, May 2018, pp. 1–5.

[88] D. Fayard and G. Plateau, "Resolution of the 0–1 knapsack problem: Comparison of methods," *Math. Program.*, vol. 8, no. 1, pp. 272–307, Dec. 1975.

[89] S. Martello and P. Toth, "A new algorithm for the 0–1 knapsack problem," *Manag. Sci.*, vol. 34, no. 5, pp. 633–644, May 1988.

[90] H. Chen, R. Abbas, P. Cheng, M. Shirvanimoghaddam, W. Hardjawana, W. Bao, Y. Li, and B. Vucetic, "Ultra-reliable low latency cellular networks: Use cases, challenges and approaches," *IEEE Commun. Mag.*, vol. 56, no. 12, pp. 119–125, Dec. 2018.

[91] S. R. Pokhrel and J. Choi, "Improving TCP performance over WiFi for Internet of Vehicles: A federated learning approach," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 6798–6802, Jun. 2020.

[92] S. R. Pokhrel and J. Choi, "Low-delay scheduling for Internet of Vehicles: Load-balanced multipath communication with FEC," *IEEE Trans. Commun.*, vol. 67, no. 12, pp. 8489–8501, Dec. 2019.

[93] J. Cloud and M. Medard, "Multi-path low delay network codes," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–7.

[94] A. Cohen, R. G. L. D'Oliveira, S. Salamatian, and M. Médard, "Network coding-based post-quantum cryptography," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 1, pp. 49–64, Mar. 2021.

[95] R. J. McEliece, "A public-key cryptosystem based on algebraic," *Coding Thv*, vol. 4244, pp. 114–116, Apr. 1978.

[96] K. Tsiknas and G. Stamatelos, "Comparative performance evaluation of TCP variants in Wimax (and WLANs) network configurations," *J. Comput. Netw. Commun.*, vol. 2012, pp. 1–9, Jan. 2012.

[97] C. Bormann, M. Ersue, and A. Keranen, "Terminology for constrained-node networks," Internet Eng. Task Force, 2014.

[98] M. Mathis and J. Mahdavi, "Forward acknowledgement: Refining TCP congestion control," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 26, no. 4, pp. 281–291, 1996.

[99] N. V. Ha and M. Tsuru, "TCP/NC performance in bi-directional loss environments," in *Proc. Int. Conf. Electron., Inf., Commun. (ICEIC)*, Jan. 2019, pp. 1–4.

[100] J.-W. Huang, K.-C. Yang, H.-Y. Hsieh, and J.-S. Wang, "Transmission control for fast recovery of rateless codes," *Int. J. Adv. Comput. Sci. Appl.*, vol. 4, no. 3, pp. 26–30, 2013.

[101] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 4, pp. 56–67, Oct. 1998.

[102] N. Bonello, Y. Yang, S. Aissa, and L. Hanzo, "Myths and realities of rateless coding," *IEEE Commun. Mag.*, vol. 49, no. 8, pp. 143–151, Aug. 2011.

[103] N. Bonello, R. Zhang, S. Chen, and L. Hanzo, "Reconfigurable rateless codes," *IEEE Trans. Wireless Commun.*, vol. 8, no. 11, pp. 5592–5600, Nov. 2009.

[104] A. S. Abdullah, M. J. Abbasi, and N. Fisal, "Review of rateless-network-coding-based packet protection in wireless sensor networks," *Mobile Inf. Syst.*, vol. 2015, pp. 1–13, Jul. 2015.

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

[106] 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.

[107] *Study on Physical Layer Enhancements for NR Ultra-Reliable and Low Latency Case (URLLC), (Release 16)*, document TR 38.824, V16. 0.0, 3GPP, 2019.

[108] 5G Americas, "New services & applications with 5G ultra-reliable low latency communications," 2018. [Online]. Available: https://www.5gamericas.org/wp-content/uploads/2019/07/5G_Americas_URLLLC_White_Paper_Final_updateJW.pdf

[109] G. Durisi, T. Koch, and P. Popovski, "Towards massive, ultra-reliable, and low-latency wireless communication with short packets," 2015, *arXiv:1504.06526*.

[110] X. Sun, S. Yan, N. Yang, Z. Ding, C. Shen, and Z. Zhong, "Short-packet downlink transmission with non-orthogonal multiple access," *IEEE Trans. Wireless Commun.*, vol. 17, no. 7, pp. 4550–4564, Jul. 2018.

[111] G. Ozcan and M. C. Gursoy, "Throughput of cognitive radio systems with finite blocklength codes," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 11, pp. 2541–2554, Nov. 2013.

[112] C. Yue, V. Miloslavskaya, M. Shirvanimoghaddam, B. Vucetic, and Y. Li, "Efficient decoders for short block length codes in 6G URLLC," *IEEE Commun. Mag.*, vol. 61, no. 4, pp. 84–90, Apr. 2023.

[113] M. R. Amini and M. W. Baidas, "Reliability-latency tradeoffs in random access ultra-reliable low-latency energy-harvesting 5G networks with finite blocklength codes," in *Proc. IEEE 31st Annu. Int. Symp. Pers., Indoor Mobile Radio Commun.*, Aug. 2020, pp. 1–6.

[114] J. Choi and J. Ding, "Network coding for K-repetition in grant-free random access," *IEEE Wireless Commun. Lett.*, vol. 10, no. 11, pp. 2557–2561, Nov. 2021.

[115] J. Choi, "Sliding network coding for URLLC," *IEEE Trans. Wireless Commun.*, vol. 21, no. 6, pp. 4424–4433, Jun. 2022.
[116] F. Gabriel, A. K. Chorppath, I. Tsokalo, and F. H. P. Fitzek, "Multipath communication with finite sliding window network coding for ultra-reliability and low latency," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2018, pp. 1–6.
[117] S. Wunderlich, F. Gabriel, S. Pandi, F. H. P. Fitzek, and M. Reisslein, "Caterpillar RLNC (CRLNC): A practical finite sliding window RLNC approach," *IEEE Access*, vol. 5, pp. 20183–20197, 2017.
[118] V. Roca, B. Teibi, C. Burdinat, T. Tran, and C. Thienot, "Block or convolutional AL-FEC codes? A performance comparison for robust low-latency communications," Unpublished Work. document HAL Open-Arch. hal-01395937, Nov. 2016.
[119] Y. Zhang, W. Zhao, P. Dong, X. Du, W. Qiao, and M. Guizani, "Improve the reliability of 6G vehicular communication through skip network coding," *Veh. Commun.*, vol. 33, Jan. 2022, Art. no. 100400.
[120] E. Tasdemir, M. Tömösközi, 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.
[121] 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.
[122] 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.
[123] 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. 24th Eur. Wireless Conf.*, May 2018, pp. 1–7.
[124] M. Tömösközi, P. Seeling, P. Ekler, and F. H. P. Fitzek, "Robust header compression version 2 power consumption on Android devices via tunnelling," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2017, pp. 418–423.
[125] S. Pandi, S. Wunderlich, and F. H. P. Fitzek, "Reliable low latency wireless mesh networks—From myth to reality," in *Proc. 15th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2018, pp. 1–2.
[126] M. Toemoeskoezi, I. Tsokalo, S. Pandi, F. H. P. Fitzek, and P. Ekler, "Header compression in opportunistic routing," in *Proc. 24th Eur. Wireless Conf.*, May 2018, pp. 1–6.
[127] M. Tomoskozi, D. Lucani, F. H. P. Fitzek, and P. Ekler, "Unidirectional robust header compression for reliable low latency mesh networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.
[128] M. Taghouti, D. E. Lucani, M. V. Pedersen, and A. Bouallegue, "Random linear network coding for streams with unequally sized packets: Overhead reduction without zero-padded schemes," in *Proc. 23rd Int. Conf. Telecommun. (ICT)*, May 2016, pp. 1–6.
[129] H. Wu, Y. Li, Y. Hu, B. Tang, and Z. Bao, "On optimizing effective rate for random linear network coding over burst-erasure relay links," *IEEE Wireless Commun. Lett.*, vol. 8, no. 2, pp. 588–591, Apr. 2019.
[130] D. E. Lucani, M. V. Pedersen, D. Ruano, C. W. Sørensen, F. H. P. Fitzek, J. Heide, O. Geil, V. Nguyen, and M. Reisslein, "Fulcrum: Flexible network coding for heterogeneous devices," *IEEE Access*, vol. 6, pp. 77890–77910, 2018.
[131] V. Nguyen, J. A. Cabrera, G. T. Nguyen, F. Gabriel, C. Lehmann, S. Mudriievskyi, and F. H. P. Fitzek, "Adaptive decoding for fulcrum codes," in *Proc. IEEE 9th Annu. Inf. Technol., Electron. Mobile Commun. Conf. (IEMCON)*, Nov. 2018, pp. 133–139.
[132] 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.
[133] N. Ma and M. Diao, "CoFi: Coding-assisted file distribution over a wireless LAN," *Symmetry*, vol. 11, no. 1, p. 71, Jan. 2019.
[134] Y. Li, S. Blostein, and W.-Y. Chan, "Systematic network coding for two-hop lossy transmissions," *EURASIP J. Adv. Signal Process.*, vol. 2015, no. 1, pp. 1–14, Dec. 2015.
[135] E. Tasdemir, V. Nguyen, G. T. Nguyen, F. H. P. Fitzek, and M. Reisslein, "FSW: Fulcrum sliding window coding for low-latency communication," *IEEE Access*, vol. 10, pp. 54276–54290, 2022.

PATRICK ENENCHE received the B.E. degree in electrical and computer engineering and the M.E. degree in communication engineering from the Federal University of Technology, Minna, Nigeria, in 2013 and 2018, respectively. He is currently pursuing the Ph.D. degree with the Department of Information and Communication Engineering, Hannam University, Daejeon, South Korea. His master's research work centered on ultraviolet absorption spectroscopy for accurate measurement of ozone concentration. His current research area revolves around network coding and ultra-reliable and low-latency communication for 5G and beyond. Notably, he is a registered and practicing Engineer with the Council for the Regulation of Engineering in Nigeria (COREN).

DONG HO KIM (Senior Member, IEEE) received the B.S. degree from Yonsei University, South Korea, in 1997, and the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology, South Korea, in 1999 and 2004, respectively. He was with the 4G Wireless Technology Laboratory, Samsung Advanced Institute of Technology from 2004 to 2006, and the Mobile Communications Research Institute, Samsung Electronics, from 2006 to 2007. Since 2007, he has been with the Department of Electronic and IT Media Engineering, Seoul National University of Science and Technology, South Korea. His current research interests include the convergence of broadcasting and mobile communication: joint source-channel coding, multimedia signal processing, and all aspects of OFDM-MIMO systems.

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 (SeoulTech), South Korea, in 2012, 2014, and 2018, respectively. He was a Senior Researcher with the Deutsche Telekom Chair of Communication Networks, Technical University Dresden (TU Dresden), from 2018 to 2021. He is currently an Assistant Professor with Hannam University, South Korea. His current research interests include 5G communication networks and services for robust and reliable multimedia communications.